

# НТО

## МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников  
«Национальная технологическая олимпиада»

по профилю  
«Автономные транспортные системы»

2021/22 учебный год

<http://ntcontest.ru>

# Оглавление

<b>1 Профиль «Автономные транспортные системы»</b>	<b>5</b>
<b>I Первый отборочный этап</b>	<b>8</b>
<b>I.1 Задачи первого этапа. Информатика</b>	<b>8</b>
I.1.1 Первая попытка. Задачи 8–11 класса . . . . .	8
I.1.2 Вторая попытка. Задачи 8–11 класса . . . . .	17
I.1.3 Третья попытка. Задачи 8–11 класса . . . . .	24
I.1.4 Четвертая попытка. Задачи 8–11 класса . . . . .	33
<b>I.2 Задачи первого этапа. Физика</b>	<b>44</b>
I.2.1 Первая попытка. Задачи 8–9 класса . . . . .	44
I.2.2 Первая попытка. Задачи 10–11 класса . . . . .	46
I.2.3 Вторая попытка. Задачи 8–9 класса . . . . .	49
I.2.4 Вторая попытка. Задачи 10–11 класса . . . . .	51
I.2.5 Третья попытка. Задачи 8–9 класса . . . . .	54
I.2.6 Третья попытка. Задачи 10–11 класса . . . . .	56
I.2.7 Четвертая попытка. Задачи 8–9 класса . . . . .	59
I.2.8 Четвертая попытка. Задачи 10–11 класса . . . . .	61
<b>II Второй отборочный этап</b>	<b>65</b>
<b>II.1 Индивидуальная часть</b>	<b>65</b>
<b>II.2 Командная часть</b>	<b>74</b>
<b>III Заключительный этап</b>	<b>85</b>
<b>III.1 Индивидуальный предметный тур</b>	<b>85</b>
III.1.1 Информатика. 8–11 класс . . . . .	85
III.1.2 Физика. 8–9 классы . . . . .	97
III.1.3 Физика. 10–11 классы . . . . .	109
<b>III.2 Командный практический тур</b>	<b>122</b>

III.2.1 Требования к команде . . . . .	122
III.2.2 Оборудование и программное обеспечение . . . . .	122
III.2.3 Описание задачи . . . . .	123
<b>IV Критерии определения победителей и призеров</b>	<b>151</b>

# Профиль «Автономные транспортные системы»

Профиль «Автономные транспортные системы» посвящен применению технологий искусственного интеллекта в сфере беспилотного транспорта. Финалистам предстоит создать и запустить полностью автономную мультимодальную транспортную систему для доставки товара с фабрики производителя до конечного покупателя без вмешательства человека. Беспилотный автомобиль везет груз на склад через весь город, соблюдая правила дорожного движения. На складе товар проходит автоматизированную сортировку, подается на квадрокоптер и доставляется до конечного покупателя дроном в режиме автономного полета.

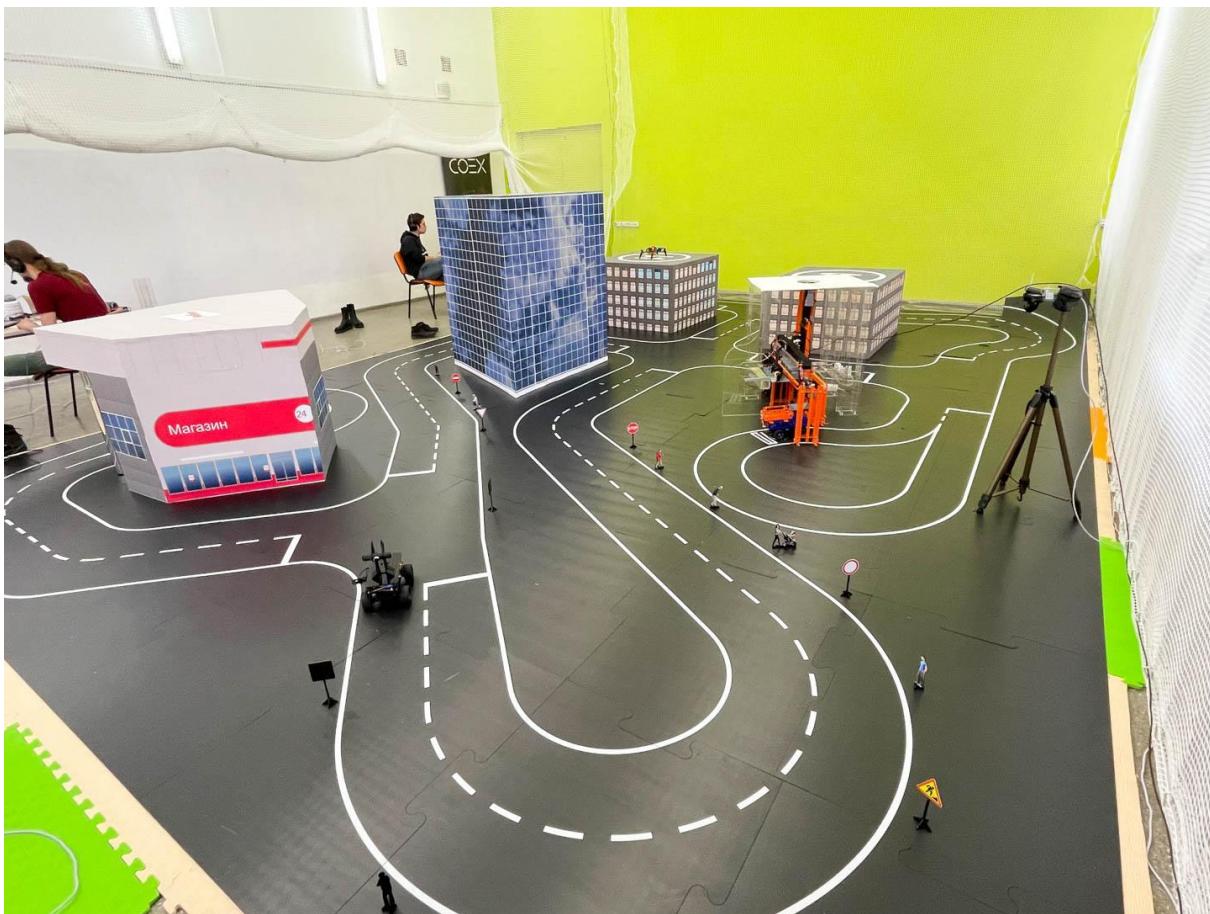
Сквозная технология профиля — **компьютерное зрение**. В течение всего года участники учатся применять его в различных отраслях:

- локальное позиционирование беспилотного автомобиля и детектирование объектов городской среды: дорожных знаков, сигналов светофоров, приближающихся пешеходов, разметки — и реагирование на них с соблюдением правил дорожного движения;
- навигация квадрокоптера по графическим меткам и различным визуальным маркерам;
- распознавание маркировки для решения задач промышленной робототехники на сортировочных конвейерных лентах.

Организаторы профиля: ООО «Академия Высоких Технологий», Московский политехнический университет и ГК «Геоскан».

В первом отборочном дистанционном этапе (индивидуальный зачет) определяется общий уровень подготовки участников по школьным предметам: математика и физика. Задачи по математике проверяют у участников знания по алгебре, комбинаторике, геометрии. Задачи по физике относятся к разделам механики, оптики и электричества.

В ходе решения задач школьники должны продемонстрировать понимание принципов работы электрических схем, двигателей постоянного тока, взаимодействия шасси автомобиля и дорожного полотна, формирование изображения в видеокамере. Количество попыток сдачи решения задач не ограничено. Таким образом, в ходе первого отборочного этапа участники профиля «Автономные транспортные системы» получают основополагающее понимание по тематике и технологиям профиля и готовятся не только к решению задач следующего этапа, но и расчету физических процессов задачи заключительного этапа.



Полигон профиля «Автономные транспортные системы»

Второй отборочный этап посвящен формированию навыков программирования, работе с алгоритмами компьютерного зрения для детектирования объектов и распознавания образов, а также обучению нейронных сетей. В ходе решения задач второго этапа учащиеся сталкиваются с отдельными элементами комплексной задачи заключительного этапа и, таким образом, могут отработать ее ключевые узлы заранее.

Эта работа помогает участникам лучше понять сильные стороны каждого члена команды и наиболее эффективно распределить обязанности и зоны ответственности.

Организаторы профиля предоставляют участникам онлайн-курс по применению компьютерного зрения с глубоким обучением, проводят регулярные консультации с участниками в формате онлайн и отвечают на возникающие вопросы в чате участников в Telegram и в сообществе в Discord.

Соревнования заключительного этапа проходили в распределенном формате. Задача участников состояла в том, чтобы создать и запустить мультимодальную транспортную систему для доставки товара с фабрики производителя до конечного покупателя без вмешательства человека. Беспилотный автомобиль везет груз на склад через весь город, детектирует встречающиеся на пути объекты городской среды: дорожные знаки, светофоры, пешеходов, разметку — и реагирует на них с соблюдением правил дорожного движения. На складе товар проходит автоматизированную сортировку по графическим меткам, напечатанным на стороне коробки, и подается на квадрокоптер. Дрон в режиме автономного полета доставляет груз до конечного покупателя и отправляется на дозаправку.



Операторы оборудования на площадке заключительного этапа олимпиады

# 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*

<b>Стандартный ввод</b>
810
<b>Стандартный вывод</b>
64.8

*Пример №2*

<b>Стандартный ввод</b>
1530
<b>Стандартный вывод</b>
100

*Пример №3*

<b>Стандартный ввод</b>
10000
<b>Стандартный вывод</b>
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  $a_nx^n + a_{n-1}x^{n-1} + \dots + a_2x^2 + a_1x + a_0$ . От вас требуется написать программу, которая найдет произведение этого многочлена на  $x + 1$ . Многочлен задан своими коэффициентами  $a_n, a_{n-1}, \dots, a_2, a_1, a_0$ . 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.  $1 \leq n \leq 100$ . Далее во второй строке через пробел подается  $n + 1$  целое число — коэффициенты многочлена  $a_n, a_{n-1}, \dots, a_2, a_1, a_0$ . 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*

<b>Стандартный ввод</b>
3
2 3 0 1
<b>Стандартный вывод</b>
2 5 3 1 1

### **Решение**

Найдем произведение многочлена  $a_nx^n + a_{n-1}x^{n-1} + \dots + a_2x^2 + a_1x + a_0$  и  $x + 1$ .

$$\begin{aligned}
 & (a_nx^n + a_{n-1}x^{n-1} + \dots + a_2x^2 + a_1x + a_0)(x + 1) = \\
 & = (a_nx^{n+1} + a_{n-1}x^n + \dots + a_2x^3 + a_1x^2 + a_0x) + \\
 & + (a_nx^n + a_{n-1}x^{n-1} + \dots + a_2x^2 + a_1x + a_0) = \\
 & = a_nx^{n+1} + (a_{n-1} + a_n)x^n + \dots + (a_1 + a_2)x^2 + (a_0 + a_1)x + a_0
 \end{aligned}$$

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=' ')

```

---

```

5  for t in x:
6      print(t+prev,end=' ')
7      prev=t;
8  print(prev)

```

Ниже представлено решение в функциональном стиле на языке Python 3.

```

1 x=list(map(int,input().split()))
2 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.  $2 \leq n \leq 30$ . 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.

```

1 n=int(input())
2 nxt=dict()
3 for i in range(n-1):
4     val,key=input().split()
5     nxt[key]=val
6 cur=next(iter(nxt.keys()-nxt.values()))
7 print(cur)
8 for i in range(n-1):
9     cur=nxt[cur]
10    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*

<b>Стандартный ввод</b>
3 9 5 3 7
<b>Стандартный вывод</b>
7
3 5
9 3

#### *Пример №2*

<b>Стандартный ввод</b>
3 9 5 1 7
<b>Стандартный вывод</b>
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.

```

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

```

Ниже представлено решение на языке Python 3 с сортировкой.

```

1  x=sorted(list(map(int,input().split())),reverse=True)
2  if x[0]<x[1]+x[2] and x[2]<x[3]+x[4]:
3      print(x[2])
4      print(x[0],x[1])
5      print(x[3],x[4])
6  elif x[0]<x[1]+x[3] and x[1]<x[2]+x[4]:
7      print(x[1])
8      print(x[0],x[3])
9      print(x[2],x[4])
10 elif x[0]<x[1]+x[4] and x[1]<x[2]+x[3]:
11     print(x[1])
12     print(x[0],x[4])
13     print(x[2],x[3])
14 else:
15     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 расселились 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.  $1 \leq n \leq 300000$ . Во второй строке через пробел записаны  $n$  натуральных чисел  $r_1, \dots, r_n$  — ранги всех рыцарей.  $r_i \leq 10^6$ ;  $r_i \neq r_{i+1}$ ;  $r_1 \neq r_n$ . В третьей строке через пробел записаны два натуральных числа  $p$  и  $t$ .  $1 \leq p \leq n$ ;  $t \leq 10^6$ . 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

<b>Стандартный ввод</b>
9
7 5 6 8 9 10 12 10 8
2 5
<b>Стандартный вывод</b>
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.

```

1 n=int(input())
2 lst=list(map(int,input().split()))
3 p,r=map(int,input().split())
4 lst=lst[p:]+lst[:p]
5 i=0
6 j=n-1
7 while i<j:
8     if lst[i]==lst[j]==r:
9         r+=2
10        i+=1
11        j-=1
12    elif lst[i]==r:
13        r+=1
14        i+=1
15    elif lst[j]==r:
16        r+=1
17        j-=1
18    else:
19        break
20 if i==j and lst[i]==r:
21     i+=1
22 print(j-i+2)

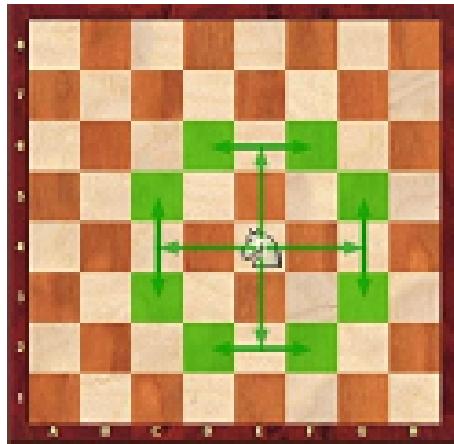
```

## 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  $8 \times 8$  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*

<b>Стандартный ввод</b>
5
3
<b>Стандартный вывод</b>
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.

```

1 a=int(input())
2 b=int(input())
3 ans=0
4 for i in range(1,9):
5     for j in range(1,9):
6         if a!=i and b!=j and abs(a-i)+abs(b-j)==3:
7             ans+=1
8 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  $\{3, 6, 9, 12, 15, 18, 21, 24, 27, 30, \dots\}$ . Первые 10 чисел из королевства Боба — это  $\{1, 2, 4, 5, 7, 8, 10, 11, 13, 14, \dots\}$ . 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.  $1 \leq n \leq 10^5$ . Во второй строке через пробел подается  $n$  натуральных чисел  $a_1, a_2, \dots, a_n$ . 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  $b_1, b_2, \dots, b_n$ . 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  $n \leq 10$ ,  $a_i \leq 1000$ . В следующих пяти тестах  $n \leq 10^5$ ,  $a_i \leq 10^6$ . В последних 10 тестах  $a_i \leq 10^{18}$ .

## *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.

```

1 n=int(input())
2 for x in map(int,input().split()):
3     if x%3!=0:
4         print(3*(x-x//3),end=' ')
5     else:
6         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 «а» до «з» 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*

<b>Стандартный ввод</b>
A Bra Kada Bra
<b>Стандартный вывод</b>
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.

```

1 unique=set()
2 other=''
3 for x in input().replace(' ','').lower():
4     if x in unique:
5         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 А 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  $u \neq v$ ;  $u, v \leq 50$ ;  $t, d \leq 10000$ . 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

<b>Стандартный ввод</b>
10 8 20 60
<b>Стандартный вывод</b>
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.

```

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

```

Ниже представлено решение бинарным поиском на языке Python 3.

```

1 u,v,t,d=map(int,input().split())
2 if u>v:
3     u,v=v,u
4 left=max(u,v)
5 right=200
6 for i in range(100):
7     mid=(left+right)/2
8     if t*v/(mid-v)-t*u/(mid-u)>d:
9         left=mid
10    else:
11        right=mid
12 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  $1 \leq a \leq b \leq 10^{18}$ . 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  $1 \leq a \leq b \leq 1000$ . 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.

```

1  a,b = map(int,input().split())
2  a=list(bin(a)[2:])
3  b=list(bin(b)[2:])
4  while len(a)<len(b):
5      a=['0']+a
6  for i in range(len(a)-1,-1,-1):
7      a[i]='1'
8      if a>b:
9          a[i]='0'
10         break
11 print(int(''.join(a),2))

```

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

```

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

```

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

```

1  a,b = map(int,input().split())
2  while a | (a+1) <= b:
3      a = a | (a+1)
4  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  $p$  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  $n \leq a + b$ .

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

### ***Примеры***

#### *Пример №1*

<b>Стандартный ввод</b>
59
35
40
<b>Стандартный вывод</b>
30 29

#### *Пример №2*

<b>Стандартный ввод</b>
59
41
25
<b>Стандартный вывод</b>
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.

```

1 n=int(input())
2 a=int(input())
3 b=int(input())
4 x, y = (n+1)//2, n//2
5 if x>a:
6     y+=x-a
7     x=a
8 if y>b:
9     x+=y-b
10    y=b
11 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.  $2 \leq n \leq 100$ . Во второй строке через пробел записаны  $n$  целых чисел  $a_1, \dots, a_n$ , задающих высоту каждой точки.  $-416 \leq a_i \leq 8848$ .

## **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.

```

1 n = int(input())
2 h = list(map(int, input().split()))
3 a1,a2=0,0
4 for i in range(1,n):
5     if h[i]>h[i-1]:
6         if h[i-1]>=2000:
7             a2+=h[i]-h[i-1]
8         elif h[i]<=2000:
9             a1+=h[i]-h[i-1]
10        else:
11            a1+=2000-h[i-1]
12            a2+=h[i]-2000
13 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  $3 \times 3$ . Тогда Аленка может разломить ее на две части  $3 \times 1$  и  $3 \times 2$ . 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  $3 \times 1$ , а от оставшегося куска  $3 \times 2$  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.  $1 \leq n, m \leq 100$ ;  $n + m \geq 3$ .

#### **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*

<b>Стандартный ввод</b>
3 3
<b>Стандартный вывод</b>
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.

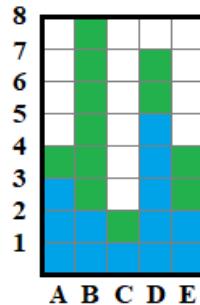
```

1 n, m = map(int, input().split())
2 k=[False]*(n*m)
3 for i in range(1,n+1):
4     for j in range(1,m+1):
5         k[n*m-i*j]=True
6 for i in range(1,n*m):
7     if k[i]:
8         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.  $2 \leq n \leq 50$ . Во второй строке через пробел записано  $n$  натуральных чисел  $a_1, \dots, a_n$  — значения первого свойства для каждой категории. В третьей строке аналогично записаны натуральные числа  $b_1, \dots, b_n$  — значения второго свойства.  $1 \leq a_i, b_i \leq 20$ .

## **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	Стандартный вывод
<pre>5 3 2 1 5 2 1 6 1 2 2</pre>	<pre>.*... .*.** .*.** .*.# **.## ##.## ##### ####</pre>

## **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()))
```

```

3  b=list(map(int,input().split()))
4  h=max([a[i]+b[i] for i in range(n)])
5  s=[[.]*n for i in range(h)]
6  for i in range(n):
7      for j in range(a[i]):
8          s[j][i]='#'
9      for j in range(a[i],a[i]+b[i]):
10         s[j][i]='*'
11 for x in reversed(s):
12     print(''.join(x))

```

Ниже представлено решение на языке Python 3 без двумерного массива.

```

1  n=int(input())
2  a=list(map(int,input().split()))
3  b=list(map(int,input().split()))
4  h=max([a[i]+b[i] for i in range(n)])
5  for j in range(h):
6      for i in range(n):
7          if h-j <= a[i]:
8              print('#',end=' ')
9          elif h-j <= a[i]+b[i]:
10             print('*',end=' ')
11         else:
12             print('.',end=' ')
13     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.  $m \leq 60$ . 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  $m \leq 10$  и  $n \leq 100$ , 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  $\alpha$  имеет длину  $k$ , меньшую чем  $m$ , то любое слово, полученное приписыванием к  $\alpha$  справа  $m - k$  символов, будет недоступно. Существует  $2^{m-k}$  способов дописать справа к  $\alpha$  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.

```

1 n,m=map(int,input().split())
2 ans=2**m
3 wset=set()
4 for x in input().split():
5     if len(x)>m:
6         wset.add(x[:m])
7     else:
8         ans-=2**(m-len(x))
9 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.  $5 \leq u \leq 50$ .

#### *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*

<b>Стандартный ввод</b>
12.5
<b>Стандартный вывод</b>
4 48

##### *Пример №2*

<b>Стандартный ввод</b>
15.03
<b>Стандартный вывод</b>
4 0

*Пример №3*

<b>Стандартный ввод</b>
15.04
<b>Стандартный вывод</b>
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.  $1 \leq n \leq 322$ .

### ***Формат выходных данных***

Вывести требуемое предложение сказки в одной строке. Слова должны быть разделены ровно одним пробелом. Перед знаками препинания пробел не ставится. Проверка 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*

<b>Стандартный ввод</b>
1
<b>Стандартный вывод</b>
Talking horse Julius was walking along the road and met 40 robbers.

*Пример №2*

<b>Стандартный ввод</b>
2
<b>Стандартный вывод</b>
- Hello, robber number 1,- said talking horse Julius.

*Пример №3*

<b>Стандартный ввод</b>
3
<b>Стандартный вывод</b>
- Hello, talking horse Julius,- said robber number 1.

*Пример №4*

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

*Пример №5*

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

*Пример №6*

<b>Стандартный ввод</b>
162
<b>Стандартный вывод</b>
- Goodbye, robber number 1,- said talking horse Julius.

*Пример №7*

<b>Стандартный ввод</b>
163
<b>Стандартный вывод</b>
- Goodbye, talking horse Julius,- said robber number 1.

*Пример №8*

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

*Пример №9*

<b>Стандартный ввод</b>
243
<b>Стандартный вывод</b>
- Goodbye, talking horse Julius,- said horse of robber number 1.

*Пример №10*

<b>Стандартный ввод</b>
322
<b>Стандартный вывод</b>
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.

```

1 n=int(input())
2 if n==1:
3     print('Talking horse Julius was walking along the road and met 40 robbers.')
4 elif n<82 and n%2==0:
5     print('- Hello, robber number '+str(n//2)+',- said talking horse Julius.')
6 elif n<82:
7     print('- Hello, talking horse Julius,- said robber number '+str(n//2)+'.')
8 elif n<162 and n%2==0:
9     print('- Hello, horse of robber number '+str((n-80)//2)+',- said talking horse
10    ↪ Julius.')
11 elif n<162:
12     print('- Hello, talking horse Julius,- said horse of robber number
13    ↪ '+str((n-80)//2)+'.')
14 elif n<242 and n%2==0:
15     print('- Goodbye, robber number '+str((n-160)//2)+',- said talking horse
16    ↪ Julius.')
17 elif n<242:
18     print('- Goodbye, talking horse Julius,- said robber number
19    ↪ '+str((n-160)//2)+'.')
20 elif n<322 and n%2==0:
21     print('- Goodbye, horse of robber number '+str((n-240)//2)+',- said talking
22    ↪ horse Julius.')
23 elif n<322:
24     print('- Goodbye, talking horse Julius,- said horse of robber number
25    ↪ '+str((n-240)//2)+'.')
26 else:
27     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.  $2 \leq n \leq 400000$ . 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  $n \leq 100$ .

#### **Примеры**

##### *Пример №1*

<b>Стандартный ввод</b>
7
LLRLRR
<b>Стандартный вывод</b>
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, за дающей 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.

```

1 n=int(input())
2 x=input()
3 ans=''
4 for i in range(n-1,0,-1):
5     if x[i-1]=='L':
6         print(i,end=' ')
7 print(0,end=' ')
8 for i in range(1,n):
9     if x[i-1]=='R':
10        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 сеансами

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

### *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  $t \leq 1000$ . В следующих 10 тестах  $t \leq 10^6$ .

### *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.

---

```

1 n,d,s=map(int,input().split())
2 m=1
3 t=0
4 for i in range(n):
5     a,b=map(int,input().split())
6     k=(a-t)//d
7     t+=d*k
8     m+=k
9     if t+d<b:
10        t=a
11        m+=1
12 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.  $1 \leq n \leq 100$ . 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.

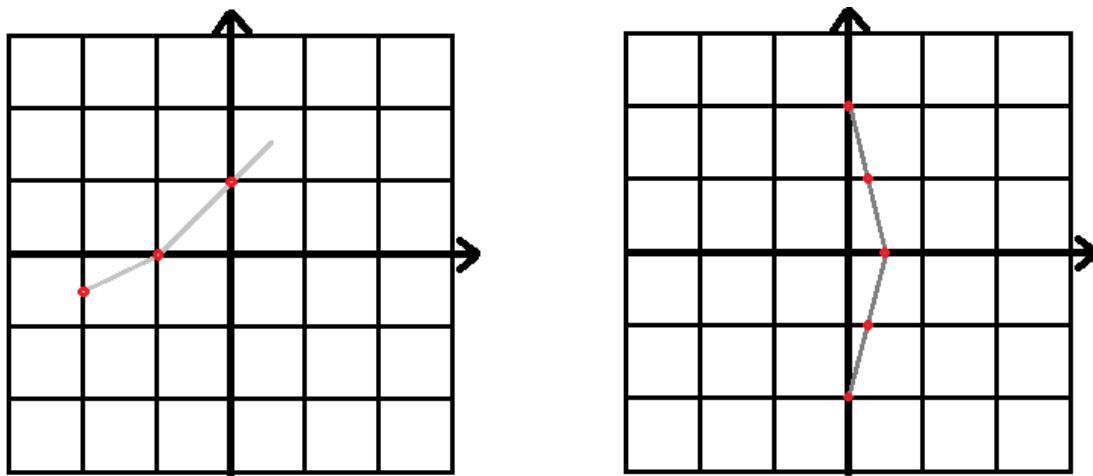
#### ***Формат выходных данных***

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

```

---

```
5     return max(0,math.floor(b-0.001)-math.ceil(a+0.001)+1)
6
7 n=int(input())
8 for i in range(n):
9     x1,y1,x2,y2=map(float,input().split());
10    print(max(bw(x1,x2),bw(y1,y2))+
11 int((x1==round(x1) or y1==round(y1))+
12 int((x2==round(x2) or y2==round(y2))))
```

# Задачи первого этапа. Физика

## Первая попытка. Задачи 8–9 класса

### Задача I.2.1.1. Электросамокат (20 баллов)

Темы: сила тока и тепло.

Емкость батареи электросамоката равна  $q = 7500 \text{ мА} \cdot \text{ч}$ , а ее внутреннее сопротивление составляет  $r = 0,05 \text{ Ом}$ . Какое тепло выделится на батареи, если ее зарядить полностью постоянным током за время  $t = 6 \text{ ч}$ ?

Ответ дать с точностью до десятых долей кДж.

#### *Решение*

Тепло равно:

$$Q = I^2 rt = \left(\frac{q}{t}\right)^2 rt = \frac{rq^2}{t} = 1,7 \text{ кДж.}$$

Точность 0,1 кДж.

Ответ:  $1,7 \pm 0,1$ .

### Задача I.2.1.2. Электросамокат (20 баллов)

Темы: мощность электромотора.

Продолжение задачи I.2.1.1.

Максимальный ток разрядки батареи электросамоката в десять раз превышает средний ток зарядки батареи. Какова максимальная мощность электромотора самоката, если постоянное напряжение батареи равно  $U = 24 \text{ В}$ ?

Ответ дать с точностью до 1 Вт.

#### *Решение*

Максимальная мощность равна:

$$P = 10IU = 10U \frac{q}{t} = 300 \text{ Вт.}$$

Точность 1 Вт.

Ответ:  $300 \pm 1$ .

### Задача I.2.1.3. Электросамокат (20 баллов)

Темы: скорость и мощность.

Продолжение задач I.2.1.1 и I.2.1.2.

С какой максимальной скоростью может ехать человек на электросамокате по горизонтальной дороге, если сила сопротивления воздуха, действующая на человека, при этой скорости равна  $F = 30$  Н. Силой трения качения пренебречь.

Ответ дать с точностью до 1 м/с.

### *Решение*

Максимальная скорость равна:

$$V_m = \frac{P}{F} = 10 \text{ м/с.}$$

Точность 1 м/с.

**Ответ:**  $10 \pm 1$ .

### **Задача I.2.1.4. Электросамокат (20 баллов)**

Темы: закон Ома и сила сопротивления.

Продолжение задач I.2.1.1, I.2.1.2 и I.2.1.3.

Среднее сопротивление электрической цепи двигателя электросамоката равно  $R = 8,0$  Ом. Какое расстояние проедет с постоянной скоростью электросамокат по горизонтальной дороге, израсходовав весь заряд батареи? Силу сопротивления воздуха считать прямо пропорциональной скорости.

Ответ дать с точностью до 1 км.

### *Решение*

Средняя мощность двигателя равна механической мощности на скорости  $V$ :

$$\frac{U^2}{R} = F \frac{V}{V_m} V.$$

Время движения самоката равно:

$$t = q \frac{R}{U}.$$

Отсюда путь равен:

$$S = Vt = q \sqrt{\frac{RV_m}{F}} = 44 \text{ км.}$$

Точность 1 км.

**Ответ:**  $44 \pm 1$ .

### **Задача I.2.1.5. Электросамокат (20 баллов)**

*Темы: сила тяжести.*

Продолжение задач I.2.1.1, I.2.1.2, I.2.1.3 и I.2.1.4.

После полной перезарядки батареи человек массой 50 кг поднимается с постоянной скоростью на электросамокате массой 12 кг в гору высотой  $h = 50$  м на  $S = 1$  км пути. Среднее сопротивление электрической цепи двигателя электросамоката равно  $R = 8,0$  Ом. Найти эту скорость электросамоката.

Ответ дать с точностью до 0,1 м/с.

Ускорение свободного падения  $g = 10$  Н/кг.

#### **Решение**

Средняя мощность двигателя равна механической мощности на скорости  $V$ :

$$\frac{U^2}{R} = F \frac{V}{V_m} V + (m + M)g \frac{h}{S/V}.$$

Отсюда получаем квадратное уравнение для скорости  $V$ :

$$\frac{F}{V_m} V^2 + (m + M)g \frac{h}{S} V - \frac{U^2}{R} = 0.$$

Его решение:  $V = 2,0$  м/с.

Точность 0,1 м/с.

**Ответ:**  $2,0 \pm 0,1$ .

### **Первая попытка. Задачи 10–11 класса**

#### **Задача I.2.2.1. Гепард на охоте (20 баллов)**

*Темы: равнотекущенное движение.*

Гепард начинает бросок за жертвой из состояния покоя и бежит по кривой линии с постоянным тангенциальным ускорением  $a_0$  промежуток времени  $t_1 = 3,0$  с. Какой путь преодолеет гепард, чтобы достичь к концу промежутка времени значения скорости  $V_0 = 70$  км/ч?

Ответ дать с точностью до 1 м.

#### **Решение**

Путь равен:

$$S = \frac{V_0 t_1}{2} = 29 \text{ м.}$$

Точность 1 м.

**Ответ:**  $29 \pm 1$ .

**Задача I.2.2.2. Гепард на охоте (20 баллов)***Темы: ускорение.*

Продолжение задачи I.2.2.1.

Найти радиус окружности, по которой бежит гепард в момент времени 2,0 с. Тангенциальное ускорение равно центростремительному ускорению в этот момент.

Ответ дать с точностью до 1 м.

***Решение***В момент времени  $t$  скорость равна  $V = a_0 t$ .

Радиус окружности равен:

$$R = \frac{V^2}{a_0} = \frac{V_0 t^2}{a_0} = \frac{V_0 t^2}{t_1} = 26 \text{ м.}$$

Точность 1 м.

**Ответ:**  $26 \pm 1$ .**Задача I.2.2.3. Гепард на охоте (20 баллов)***Темы: скорость.*

Продолжение задач I.2.2.1 и I.2.2.2.

После времени  $t_1 = 3,0$  с гепард начинает уставать и его тангенциальное ускорение уменьшается по линейному закону  $a = a_0 - kt$ , где коэффициент  $k = a_0^2/V_0$  и время  $t$  отсчитывается с момента времени  $t_1$ . Найти максимальную скорость гепарда.

Ответ дать с точностью до 1 км/ч.

***Решение***

В момент времени  $t = \frac{a_0}{k} = t_1$  ускорение равно нулю и скорость достигает максимального значения равного:

$$V = V_0 + a_0 t - k \frac{t^2}{2} = \frac{3}{2} V_0 = 105 \text{ км/ч.}$$

Точность 1 км/ч.

**Ответ:**  $105 \pm 1$ .**Задача I.2.2.4. Гепард на охоте (20 баллов)***Темы: путь.*

Продолжение задач I.2.2.1, I.2.2.2 и I.2.2.3.

Гепард промахивается и постепенно останавливается. Какой путь пробежит гепард с момента времени  $t_1$  до остановки, если ускорение гепарда изменяется по линейному закону из задачи I.2.2.3 и после остановки становится равным нулю?

Ответ дать с точностью до 1 м.

### *Решение*

В момент времени  $t_2 = (1 + \sqrt{3}) t_1$  гепард останавливается. Тогда путь гепарда до остановки равен:

$$S = V_0 t_0 + a_0 \frac{t_2^2}{2} - k \frac{t_2^3}{6} = V_0 t_1 \left(1 + \sqrt{3}\right) \left[1 + \frac{(1 + \sqrt{3})}{2} - \frac{(1 + \sqrt{3})^2}{6}\right] = 179 \text{ м.}$$

Точность 1 м.

**Ответ:**  $179 \pm 1$ .

### *Задача I.2.2.5. Гепард на охоте (20 баллов)*

*Темы:* мощность.

Продолжение задач I.2.2.1, I.2.2.2, I.2.2.3 и I.2.2.4.

Найти максимальную удельную мощность, развиваемую гепардом за все время охоты.

Ответ дать с точностью до 1 Вт/кг.

### *Решение*

Удельная мощность гепарда равна произведению горизонтальной силы на скорость отнесенному к массе гепарда:

$$\frac{P}{m} = a(t)V(t).$$

При движении с уменьшающимся ускорением удельная мощность изменяется по закону:

$$\frac{P}{m} = a(t)V(t) = (a_0 - kt) \left(V_0 + a_0 t - \frac{kt^2}{2}\right).$$

Причем максимальная мощность достигается в момент времени  $t_1$  после начала движения гепарда ( $t = 0$ ):

$$\frac{P_{max}}{m} = a_0 V_0 = \frac{V_0^2}{t_1} = 126 \text{ Вт/кг.}$$

Точность 1 Вт/кг.

**Ответ:**  $126 \pm 1$ .

## Вторая попытка. Задачи 8–9 класса

### *Задача I.2.3.1. Айсберг (10 баллов)*

*Темы:* плотность.

Столообразный айсберг, плавающий в южной полярной области, ограничен двумя одинаковыми горизонтальными основаниями: надводным и подводным. Айсберг имеет ровные вертикальные боковые стенки и формой похож на кусок мороженого между двумя вафлями. Площадь каждого основания айсберга равна  $S = 100 \text{ км}^2$ , а его толщина составляет  $H = 200 \text{ м}$ . Айсберг состоит из пресного льда плотностью  $\rho_1 = 920 \text{ кг}/\text{м}^3$ , в толще которого находятся пузырьки сжатого воздуха, занимающие 10% объема айсберга. Плотность сжатого воздуха равна  $\rho_2 = 1,5 \text{ кг}/\text{м}^3$ . Найти массу айсберга  $m_0$  с точностью до 1 миллиона тонн.

Ответ дать в миллионах тонн.

#### *Решение*

Масса айсберга равна:

$$m_0 = \rho_1 V_1 + \rho_2 V_2 = (0,9\rho_1 + 0,1\rho_2)SH = 16563 \text{ миллионов тонн.}$$

Точность 1 миллион тонн.

Ответ:  $16563 \pm 1$ .

### *Задача I.2.3.2. Айсберг (15 баллов)*

*Темы:* сила Архимеда.

Продолжение задачи I.2.3.1.

Айсберг плавает в морской воде плотностью  $\rho = 1025 \text{ кг}/\text{м}^3$ . Найти высоту  $h$  его надводной части с точностью до десятой доли метра.

#### *Решение*

Сила тяжести равна силе Архимеда. Отсюда высота надводной части равна:

$$h = H - \frac{m_0}{\rho S} = 38,4 \text{ м.}$$

Точность 0,1 м.

Ответ:  $38,4 \pm 0,1$ .

### **Задача I.2.3.3. Айсберг (20 баллов)**

**Темы:** сила тока.

Продолжение задач [I.2.3.1](#) и [I.2.3.2](#).

По всей площади надводного основания айсберга скопилась тонкая прослойка морской воды. Между надводным основанием айсберга и его горизонтальным сечением на уровне поверхности моря существует электрическое напряжение  $U = 130$  В. Найти силу тока, протекающего в надводной части айсберга между основанием и сечением. Считать, что линии электрического тока вертикальны. Удельное сопротивление вещества айсберга равно  $\rho_0 = 5 \cdot 10^6$  Ом · м.

Ответ дать с точностью до 1 А.

#### **Решение**

Сопротивление надводной части айсберга равно  $R = \frac{\rho_0 h}{S}$ , и силу тока находим из закона Ома:

$$I = \frac{US}{\rho_0 h} = 68 \text{ А.}$$

Точность 1 А.

**Ответ:**  $68 \pm 1$ .

### **Задача I.2.3.4. Айсберг (30 баллов)**

**Темы:** тепловой баланс.

Продолжение задач [I.2.3.1](#) и [I.2.3.2](#).

Подводная часть айсберга нагревается теплой морской водой и медленно тает. Рассмотрим этот процесс. Вокруг айсберга образовалась масса  $m_1$  пресной воды при температуре  $t_1 = 0,0^\circ\text{C}$ . В результате теплового контакта с теплой морской водой массы  $m_2$  и температурой  $t$  установилось тепловой равновесие и температура пресной воды массой  $m_1$  повысилась до  $t_2 = 4,0^\circ\text{C}$ . Благодаря течению к пресной воде с массой  $m_1$  поступает следующая порция теплой морской воды массы  $m_2$  с температурой  $t$  и устанавливается новое тепловое равновесие с температурой  $t_3 = 6,0^\circ\text{C}$ . Найти температуру  $t$  теплой морской воды с точностью до  $0,1^\circ\text{C}$ . Считать, что морская вода с пресной водой не перемешиваются и они имеют разные теплоемкости.

#### **Решение**

Уравнения теплового баланса для первого контакта:

$$m_1 C_1 (t_2 - t_1) = m_2 C_2 (t - t_2)$$

и для второго контакта:

$$m_1 C_1 (t_3 - t_2) = m_2 C_2 (t - t_3).$$

Из уравнений находим температуру  $t$  теплой морской воды:

$$t = \frac{t_1 t_3 - t_2^2}{t_3 - 2t_2 + t_1} = 8,0^\circ C.$$

Точность  $0,1^\circ C$ .

**Ответ:**  $8,0^\circ C \pm 0,1^\circ C$ .

### Задача I.2.3.5. Айсберг (25 баллов)

*Темы: теплообмен.*

Продолжение задачи I.2.3.1.

Процесс таяния айсберга приводит к тому, что его масса  $m(\tau)$  уменьшается со временем  $\tau$  по закону:  $m(\tau) = m_0 + b\tau - a\tau^2$ , где константы  $a$  и  $b$  связаны с начальной массой айсберга  $m_0$  и характерным временем  $T = 1$  год соотношениями:  $b = aT$  и  $a = \frac{m_0}{4T^2}$ . Найти время жизни айсберга (полностью растает) с точностью до 0,1 года.

#### Решение

Когда айсберг полностью растает, то его масса обратится в ноль:

$$m_0 + b\tau - a\tau^2 = 0 \text{ и } T\tau - 4T^2 = 0.$$

Тогда время жизни айсберга равно:

$$\tau_{ж} = 2,6T = 2,6 \text{ года.}$$

Точность 0,1 года.

**Ответ:**  $2,6 \pm 0,1$ .

## Вторая попытка. Задачи 10–11 класса

### Задача I.2.4.1. Гонки по вертикали (10 баллов)

*Темы: динамика материальной точки.*

В мотоаттракционе «Гонки по вертикальной стене» человек на мотоцикле движется по внутренней стороне вертикального деревянного цилиндра радиуса  $R = 6$  м и высотой  $H = 8$  м. Коэффициент трения скольжения резиновой шины по дереву равен  $\mu = 0,6$ . Ускорение свободного падения  $g = 10$  Н/кг. В задачах I.2.4.1, I.2.4.2 и I.2.4.3 считать мотоцикл с человеком материальной точкой. Найти скорость движения мотоциклиста  $V$  в горизонтальной плоскости, если она в полтора раза превышает минимально возможную скорость.

Ответ дать с точностью до 1 м/с.

*Решение*

Из второго закона Ньютона минимально возможная скорость равна  $V_0 = \sqrt{gR/\mu}$ . Тогда скорость мотоциклиста  $V = 1,5\sqrt{gR/\mu} = 15$  м/с.

Точность 1 м/с.

**Ответ:**  $15 \pm 1$ .

**Задача I.2.4.2. Гонки по вертикали (15 баллов)**

*Темы:* КПД.

Найти КПД двигателя мотоцикла, если он движется со скоростью из задачи I.2.4.1. Мощность двигателя мотоцикла составляет 12 л. с. Расход бензина равен 6 л на 100 км пути. Удельная теплота сгорания бензина равна 44 МДж/кг, а плотность бензина составляет  $730$  кг/м<sup>3</sup>.

Ответ дать с точностью до 1%.

*Решение*

КПД равен отношению мощности двигателя к мощности сгорания бензина:

$$\eta = \frac{P}{Q_{уд}\rho V q_{расход}} = 31\%.$$

Точность 1%.

**Ответ:**  $31 \pm 1$ .

**Задача I.2.4.3. Гонки по вертикали (20 баллов)**

*Темы:* путь.

Продолжение задачи I.2.4.1.

Найти путь мотоцикла от основания до вершины цилиндра по винтовой линии с шагом между витками (по вертикали)  $h = 0,5$  м. Параметры цилиндра даны в задаче I.2.4.1.

Ответ дать с точностью до 1 м.

*Решение*

Путь равен длине одного витка умноженной на число витков.

$$S = \frac{H}{h} \sqrt{(2\pi R)^2 + h^2} = 603 \text{ м.}$$

Точность 1 м.

**Ответ:**  $603 \pm 1$ .

### **Задача I.2.4.4. Гонки по вертикали (25 баллов)**

*Темы: момент силы.*

Продолжение задачи I.2.4.1.

Заменив человека с мотоциклом на однородный тонкий равнобедренный треугольник, скользящий углами основания по внутренней поверхности цилиндра, найти угол наклона  $\alpha$  треугольника к горизонтальной плоскости. Высота треугольника, проведенная к его основанию равна  $a = 1,7$  м, а основание треугольника шириной  $b = 1,8$  м расположено горизонтально. Скорость углов основания треугольника равна скорости мотоцикла из задачи I.2.4.1. Ускорение свободного падения  $g = 10$  Н/кг.

Ответ дать с точностью до десятой доли градуса.

#### **Решение**

Центр масс однородного тонкого треугольника находится на его высоте на расстоянии  $a/3$  от основания и движется по окружности радиуса:

$$r = \sqrt{R^2 - \left(\frac{b}{2}\right)^2} - \frac{a}{3} \cos \alpha,$$

с угловой скоростью  $\omega = V/R$ . Запишем равенство моментов центробежной силы и силы тяжести относительно основания треугольника:

$$m\omega^2 r \frac{a}{3} \sin \alpha = mg \frac{a}{3} \cos \alpha.$$

Отсюда получаем уравнение для угла  $\alpha$ :

$$\operatorname{tg} \alpha = \frac{gR^2}{V^2 \left[ \sqrt{R^2 - \left(\frac{b}{2}\right)^2} - \frac{a}{3} \cos \alpha \right]}.$$

Так как второе слагаемое в знаменателе в десять раз меньше первого, то уравнение можно решить методом последовательных приближений.

Тогда угол равен  $\alpha = 16,5^\circ$ .

Точность  $0,2^\circ$ .

**Ответ:**  $16,5^\circ \pm 0,2^\circ$ .

### **Задача I.2.4.5. Гонки по вертикали (30 баллов)**

*Темы: центробежная сила.*

Продолжение задач I.2.4.1 и I.2.4.4.

Найти силу давления одного угла основания треугольника на поверхность цилиндра. Масса треугольника равна массе мотоцикла с человеком  $m = 250$  кг. Ускорение свободного падения  $g = 10$  Н/кг.

Ответ дать с точностью до десятой доли кН.

***Решение***

Центробежная сила уравновешивает удвоенную проекцию силы реакции вертикальной стенки. По третьему закону Ньютона сила давления равна силе реакции:

$$m\omega^2 r = 2N \frac{\sqrt{R^2 - \left(\frac{b}{2}\right)^2}}{R}.$$

Отсюда:

$$N = \frac{mV^2}{2R} \left( 1 - \frac{a \cos \alpha}{3\sqrt{R^2 - \left(\frac{b}{2}\right)^2}} \right) = 4,3 \text{ кН.}$$

Точность 0,1 кН.

**Ответ:**  $4,3 \pm 0,1$ .

**Третья попытка. Задачи 8–9 класса*****Задача I.2.5.1. Корпус атомного реактора (10 баллов)***

Темы: равномерное движение.

Первый корпус современного атомного реактора ВВЭР-ТОИ совершил путешествие от прокатного стана до Курской АЭС, преодолев 1500 км по воде на барже и 300 км по суше на автоплатформе. Общее время путешествия составило 150 суток. 85% этого времени корпус не двигался во время подготовок к передвижениям. Средняя скорость передвижения по воде в два раза превышает среднюю скорость на суше. Найти среднюю скорость на суше с точностью до 0,1 км/ч.

***Решение***

Используя формулы равномерного движения, получим среднюю скорость на суше:

$$V = \frac{S_1 + S_2/2}{0,15t} = 1,9 \text{ км/ч.}$$

Точность 0,1 км/ч.

**Ответ:**  $1,9 \pm 0,1$ .

***Задача I.2.5.2. Корпус атомного реактора (15 баллов)***

Темы: плавание тела.

Продолжение задачи I.2.5.1.

Масса корпуса атомного реактора с внутренним оборудованием составляет  $m = 606$  т. Баржа имеет прямоугольное днище: 50 м в длину и 18 м в ширину. Борта баржи вертикальны. На какую глубину увеличится осадка баржи при погрузке на нее корпуса реактора. Плотность речной воды  $\rho = 1000 \text{ кг/м}^3$ .

Ответ дать с точностью до сантиметра.

### *Решение*

Дополнительная сила тяжести равна дополнительной силе Архимеда. Отсюда увеличение глубины равно:

$$\Delta h = \frac{m}{\rho ab} = 67 \text{ см.}$$

Точность 1 см.

**Ответ:**  $67 \pm 1$ .

### *Задача I.2.5.3. Корпус атомного реактора (20 баллов)*

*Темы: сила давления.*

Продолжение задач [I.2.5.1](#) и [I.2.5.2](#).

Для перевозки корпуса атомного реактора по сушке использовалась автоплатформа на  $N = 192$  колесах с массой  $m_1 = 80$  т. Давление в шинах колес равно  $p = 8 \cdot 10^5$  Па. Ускорение свободного падения  $g = 10$  Н/кг. Найти площадь пятна контакта каждой шины с дорогой с точностью до  $1 \text{ см}^2$ . На автоплатформе находится корпус атомного реактора.

### *Решение*

Из формулы для силы давления получаем площадь пятна контакта каждой шины:

$$S = \frac{(m + m_1)g}{Np} = 447 \text{ см}^2.$$

Точность  $5 \text{ см}^2$ .

**Ответ:**  $447 \pm 5$ .

### *Задача I.2.5.4. Корпус атомного реактора (25 баллов)*

*Темы: центр тяжести.*

Продолжение задач [I.2.5.2](#) и [I.2.5.3](#).

На каком расстоянии  $x$  от поверхности дороги находится центр тяжести автоплатформы с лежащим на ней корпусом атомного реактора? Считать корпус сплошным цилиндром диаметром  $d = 6$  м и длиной 12 м, а автоплатформу считать однородной прямоугольной пластиной длиной 35 м, шириной  $c = 6$  м и высотой  $h = 2$  м. Цилиндр закреплен горизонтально вдоль платформы в ее середине. Массой крепежа пренебречь. Образующая цилиндра касается верхней плоскости пластины по ее оси симметрии.

Ответ дать с точностью до 0,1 м.

### *Решение*

Используя правило рычага, получим расстояние  $x$  от поверхности дороги до центра тяжести автоплатформы с корпусом атомного реактора:

$$x = \frac{m_1 h/2 + m(h + d_2/2)}{m_1 + m} = 4,5 \text{ м.}$$

Точность 0,1 м.

**Ответ:**  $4,5 \pm 0,1$ .

### *Задача I.2.5.5. Корпус атомного реактора (30 баллов)*

*Темы:* момент силы.

Продолжение задач [I.2.5.2](#), [I.2.5.3](#) и [I.2.5.4](#).

На какую максимальную высоту над дорогой можно поднять левый нижний край пластины с закрепленным цилиндром, чтобы еще не произошло их боковое опрокидывание? В этом случае пластина стоит на дороге на своем правом нижнем крае. Расстояние от левого края до правого равно  $c = 6$  м.

Ответ дать с точностью до 0,1 м.

### *Решение*

Боковое опрокидывание начнется в тот момент, когда вертикальная прямая, опущенная из центра тяжести, пройдет через правый край пластины. Тогда максимальную высоту над дорогой левого края пластины найдем из подобия прямоугольных треугольников:

$$H = \frac{c^2}{2\sqrt{x^2 + (c/2)^2}} = 3,3 \text{ м.}$$

Точность 0,1 м.

**Ответ:**  $3,3 \pm 0,1$ .

## **Третья попытка. Задачи 10–11 класса**

### *Задача I.2.6.1. Колесо автомобиля (10 баллов)*

*Темы:* газовые законы.

Колесо автомобиля состоит из колесного диска и бескамерной шины. Ширина шины  $b = 205$  мм, внутренний диаметр  $d_1 = 406$  мм, а внешний диаметр  $d_2 = 632$  мм. Давление воздуха внутри шины равно  $p = 0,25$  МПа, его молярная масса  $\mu = 29$  г/моль. Найти массу воздуха внутри шины, считая, что ее поперечное сечение имеет форму прямоугольника ширины  $b$ . Температура воздуха  $T = 20^\circ\text{C}$ . Шину считать нерастяжимой. Универсальная газовая постоянная  $R = 8,31 \text{ Дж}/(\text{моль} \cdot \text{К})$ .

Ответ дать с точностью до 1 г.

***Решение***

Объем шины прямоугольного сечения равен:

$$\nu = \pi b \left( (d_2/2)^2 - (d_1/2)^2 \right) = 0,038 \text{ м}^3.$$

Массу воздуха находим из уравнения состояния идеального газа:

$$m = \frac{\mu p \pi b \left( (d_2/2)^2 - (d_1/2)^2 \right)}{RT} = 112 \text{ г.}$$

Точность 2 г.

**Ответ:**  $112 \pm 2$ .

***Задача I.2.6.2. Колесо автомобиля (15 баллов)***

*Темы: сила давления.*

Продолжение задачи I.2.6.1.

Автомобиль стоит на горизонтальной поверхности. Вертикальная нагрузка на одно колесо равна  $F = 5000$  Н. Найти длину  $a$  пятна контакта шины с плоскостью. Считать, что пятно имеет форму прямоугольника ширины  $b$ . Боковой деформацией шины и изменением ее объема пренебречь. Шина имеет форму цилиндрического слоя с диаметрами  $d_1$ ,  $d_2$  и срезанным пятном контакта.

Ответ дать с точностью до 1 мм.

***Решение***

Приравниваем нагрузку силе давления, тогда длина пятна равна  $a = \frac{F}{pb} = 98$  мм.

Точность 1 мм.

**Ответ:**  $98 \pm 1$ .

***Задача I.2.6.3. Колесо автомобиля (20 баллов)***

*Темы: кинематика вращения.*

Продолжение задачи I.2.6.2.

Автомобиль поехал с постоянной скоростью  $V_0 = 120$  км/ч. Определить скорость верхней точки колеса, если оно катится без проскальзывания. Использовать данные о пятне контакта из задачи I.2.6.2.

Ответ дать с точностью до 0,1 м/с.

***Решение***

Скорости точек, лежащих на вертикальной оси, проходящей через центр колеса, пропорциональны расстоянию до поверхности. Отсюда скорость верхней точки

равна:

$$V = V_0 \left( 1 + \frac{1}{\sqrt{1 - \left( \frac{a}{d_2} \right)^2}} \right) = 67,1 \text{ м/с.}$$

Точность 0,1 м/с.

**Ответ:**  $67,1 \pm 0,1$ .

### **Задача I.2.6.4. Колесо автомобиля (25 баллов)**

Темы: электростатика.

Продолжение задачи I.2.6.2.

На всю внешнюю поверхность шины диаметром  $d_2$  (кроме пятна контакта) нанесли электрический заряд с постоянной поверхностной плотностью  $\sigma = 5 \cdot 10^{-8} \text{ Кл/м}^2$ . Найти напряженность электрического поля в центре колеса. Использовать данные о пятне контакта из задачи I.2.6.2. В расчетах пренебречь длиной  $a$  и шириной  $b$  пятна контакта по сравнению с диаметром  $d_2$ . Коэффициент пропорциональности  $k = 9 \cdot 10^9 \text{ Н} \cdot \text{м}^2/\text{Кл}^2$ .

Ответ дать с точностью до 1 В/м.

#### **Решение**

Напряженность электрического поля в центре кольца такая же, как от части внешней поверхности шины, опирающейся на пятно контакта. Поскольку эта часть поверхности имеет малые размеры, используем формулу для напряженности поля точечного заряда:

$$E = \frac{k\sigma ab}{(d_2/2)^2} = 90 \text{ В/м.}$$

Точность 2 В/м.

**Ответ:**  $90 \pm 2$ .

### **Задача I.2.6.5. Колесо автомобиля (30 баллов)**

Темы: газовые законы.

Продолжение задачи I.2.6.1.

В шине произошел прокол и образовалось маленькое отверстие площадью  $S = 0,012 \text{ мм}^2$ . Воздух выходит из шины очень медленно с постоянной скоростью  $u = 10 \text{ м/с}$ . Температура воздуха в шине остается постоянной  $T = 20^\circ\text{C}$  и объем шины с размерами из задачи I.2.6.1 не меняется. Найти время уменьшения давления  $p$  воздуха внутри шины в два раза.

Ответ дать с точностью до 1 часа.

***Решение***

За малое время  $\Delta t$  из отверстия выходит воздух массой  $\frac{m}{\nu} Su \Delta t$ . Тогда уменьшение массы воздуха  $m$  в шине объема  $\nu$  описывается уравнением:

$$\frac{\Delta m}{\Delta t} = -\frac{m}{\nu} Su.$$

Решение этого уравнения находим по аналогии с законом радиоактивного распада. Тогда время уменьшения массы воздуха  $m$  и давления  $p$  воздуха внутри шины в два раза равно:

$$T_{1/2} = \frac{\nu \ln 2}{Su} = 61 \text{ час.}$$

Точность 1 час.

**Ответ:**  $61 \pm 1$ .

**Четвертая попытка. Задачи 8–9 класса*****Задача I.2.7.1. Моноколесо (10 баллов)***

*Темы: источник тока.*

Емкость батареи моноколеса равна  $Q = 1110 \text{ Вт} \cdot \text{ч}$ . Какой постоянный ток будет протекать через батарею, если полное время зарядки батареи равно  $t = 6 \text{ ч}$ ? Напряжение на клеммах батареи равно  $U = 84 \text{ В}$ . Внутренним сопротивлением батареи пренебречь.

Ответ дать с точностью до десятых долей А.

***Решение***

Ток равен:

$$I = \frac{Q}{Ut} = 2,2 \text{ А.}$$

Точность 0,1 А.

**Ответ:**  $2,2 \pm 0,1$ .

***Задача I.2.7.2. Моноколесо (15 баллов)***

*Темы: мощность двигателя.*

Продолжение задачи I.2.7.1.

Максимальный ток разрядки батареи моноколеса в десять раз превышает постоянный ток зарядки батареи. Какова максимальная мощность электромотора самоката, если постоянное напряжение батареи равно  $U = 84 \text{ В}$ ?

Ответ дать с точностью до 1 Вт.

***Решение***

Максимальная мощность равна:

$$P_m = 10IU = 1850 \text{ Вт.}$$

Точность 1 Вт.

**Ответ:**  $1850 \pm 1$ .

***Задача I.2.7.3. Моноколесо (20 баллов)***

*Темы:* мощность силы.

Продолжение задач [I.2.7.1](#) и [I.2.7.2](#).

С какой максимальной скоростью может ехать человек на моноколесе по горизонтальной дороге, если сила сопротивления воздуха, действующая на человека и моноколесо, при этой скорости равна  $F = 132$  Н. Силой трения качения пренебречь.

Ответ дать с точностью до 1 км/ч.

***Решение***

Максимальная скорость равна:

$$V_m = P_m/F = 50 \text{ км/ч.}$$

Точность 2 км/ч.

**Ответ:**  $50 \pm 2$ .

***Задача I.2.7.4. Моноколесо (25 баллов)***

*Темы:* равномерное движение.

Продолжение задач [I.2.7.1](#), [I.2.7.2](#) и [I.2.7.3](#).

Человек на моноколесе по горизонтальной дороге проезжает расстояние  $S = 70$  км, израсходовав весь заряд батареи. Какое время затратит человек на это путешествие, если движение происходит с постоянной скоростью? Силу сопротивления воздуха считать прямо пропорциональной скорости.

Ответ дать с точностью до 1 мин.

***Решение***

Средняя мощность двигателя равна механической мощности на скорости  $V$ :

$$\frac{Q}{t} = F \frac{V}{V_m} V.$$

Весь путь равен  $S = Vt$ . Тогда время движения моноколеса равно:

$$t = \frac{FS^2}{V_m Q} = 193 \text{ мин.}$$

Точность 3 мин.

**Ответ:**  $193 \pm 3$ .

### **Задача I.2.7.5. Моноколесо (30 баллов)**

**Темы:** наклонная плоскость.

Продолжение задач [I.2.7.1](#), [I.2.7.2](#), [I.2.7.3](#) и [I.2.7.4](#).

После полной перезарядки батареи человек массой 100 кг поднимается с постоянной скоростью на моноколесе массой 25 кг в гору высотой  $h = 50$  м на  $l = 1$  км пути. Средняя мощность двигателя моноколеса равно  $P = 500$  Вт. Найти постоянную скорость моноколеса.

Ответ дать с точностью до 0,1 м/с.

Ускорение свободного падения  $g = 10$  Н/кг.

#### **Решение**

Средняя мощность двигателя равна механической мощности на скорости  $V$ :

$$P = F \frac{V}{V_m} V + (m + M)g \frac{h}{l/V}.$$

Отсюда получаем квадратное уравнение для  $V$ :

$$\frac{F}{V_m} V^2 + (m + M)g \frac{h}{l} V - P = 0.$$

Его решение: 4,7 м/с.

Точность 0,1 м/с.

**Ответ:**  $4,7 \pm 0,1$ .

### **Четвертая попытка. Задачи 10–11 класса**

#### **Задача I.2.8.1. Гонки на квадроцикле (10 баллов)**

**Темы:** второй закон Ньютона.

В аттракционе «Гонки по вертикальной стене» гонщик на квадроцикле движется по внутренней стороне вертикального бетонного цилиндра радиуса  $R = 7$  м и высотой  $H = 9$  м. Коэффициент трения скольжения резиновой шины по бетону равен  $\mu = 0,75$ . Ускорение свободного падения  $g = 10$  Н/кг. В задачах [I.2.8.1](#), [I.2.8.2](#) и [I.2.8.3](#) считать квадроцикл с гонщиком материальной точкой. Найти постоянную скорость движения гонщика  $V$  в горизонтальной плоскости на высоте 4 м, если она в два раза превышает минимально возможную скорость.

Ответ дать с точностью до 1 м/с.

***Решение***

Из второго закона Ньютона минимально возможная скорость равна  $V_0 = \sqrt{gR/\mu}$ . Тогда скорость мотоциклиста:

$$V = 2\sqrt{gR/\mu} = 19 \text{ м/с.}$$

Точность 1 м/с.

**Ответ:**  $19 \pm 1$ .

***Задача I.2.8.2. Гонки на квадроцикле (15 баллов)***

*Темы:* КПД.

Продолжение задачи I.2.8.1.

Найти КПД двигателя квадроцикла, если он движется со скоростью из задачи I.2.8.1. Мощность двигателя квадроцикла составляет 14 метрических лошадиных сил. Расход бензина равен 4,5 л на 100 км пути. Удельная теплота сгорания бензина равна 44 МДж/кг, а плотность бензина составляет  $730 \text{ кг/м}^3$ .

Ответ дать с точностью до 1 %.

***Решение***

КПД равен отношению мощности двигателя к мощности сгорания бензина:

$$\eta = \frac{P}{Q_{уд}\rho V q_{расход}} = 37\%.$$

Точность 1 %.

**Ответ:**  $37 \pm 1$ .

***Задача I.2.8.3. Гонки на квадроцикле (20 баллов)***

*Темы:* путь.

Продолжение задачи I.2.8.1.

Найти путь квадроцикла от основания до вершины цилиндра по винтовой линии с шагом между витками (по вертикали)  $h = 0,75 \text{ м}$ . Параметры цилиндра даны в задаче I.2.8.1.

Ответ дать с точностью до 1 м.

***Решение***

Путь равен длине одного витка умноженной на число витков.

$$S = \frac{H}{h} \sqrt{(2\pi R)^2 + h^2} = 528 \text{ м.}$$

Точность 1 м.

**Ответ:**  $528 \pm 1$ .

### *Задача I.2.8.4. Гонки на квадроцикле (25 баллов)*

*Темы: центр масс.*

Продолжение задачи I.2.8.1.

Заменим гонщика с квадроциклом на однородную прямоугольную пластину длиной  $a = 1,8$  м, шириной  $b = 1,0$  м и высотой  $c = 1,6$  м. Пластина скользит всеми четырьмя углами основания  $ab$  по внутренней поверхности цилиндра на высоте 4 м, причем ребра  $a$  горизонтальны. Найти минимально возможную скорость углов  $V_m$  основания пластины для такого скольжения.

Ответ дать с точностью до 0,3 м/с.

Ускорение свободного падения  $g = 10$  Н/кг.

#### *Решение*

Центр масс однородной прямоугольной пластины находится в ее геометрическом центре и движется по окружности радиуса:

$$r = \sqrt{R^2 - (a/2)^2} - (c/2)$$

с угловой скоростью  $\omega = V_m/R$ . На минимально возможной скорости  $V_m$  силы реакции в верхних углах основания обращаются в ноль. Тогда запишем равенство моментов центробежной силы и силы тяжести относительно горизонтальной оси, проходящей через нижние углы основания:

$$m\omega^2 r \frac{b}{2} = mg \frac{c}{2}.$$

Отсюда получаем минимально возможную скорость углов:

$$V_m = R \sqrt{\frac{gc}{rb}} = 11,3 \text{ м/с.}$$

Точность 0,3 м/с.

**Ответ:**  $11,3 \pm 0,3$ .

### *Задача I.2.8.5. Гонки на квадроцикле (30 баллов)*

*Темы: сила давления.*

Продолжение задач I.2.8.1 и I.2.8.4.

Найти силу давления одного угла основания  $ab$  на поверхность цилиндра. Масса пластины равна массе квадроцикла с гонщиком  $m = 220$  кг. Углы пластины двигаются с минимально возможной скоростью  $V_m$  из задачи I.2.8.4. Ускорение свободного падения  $g = 10$  Н/кг.

Ответ дать с точностью до десятой доли кН.

---

*Решение*

Центробежная сила уравновешивает удвоенную проекцию силы реакции вертикальной стенки в нижнем угле основания. По третьему закону Ньютона сила давления равна силе реакции:

$$m\omega^2 r = 2N \frac{\sqrt{R^2 - (a/2)^2}}{R}.$$

Отсюда:

$$N = \frac{mV_m^2}{2R} \left( 1 - \frac{c}{2\sqrt{R^2 - (a/2)^2}} \right) = 1,8 \text{ кН.}$$

Точность 0,1 кН.

**Ответ:**  $1,8 \pm 0,1$ .

# Второй отборочный этап

## Индивидуальная часть

### ***Задача II.1.1. Распознавание сигналов светофора (15 баллов)***

*Темы: программирование, компьютерное зрение, цветовые пространства, классификация.*

На финале программисту предстоит разрабатывать программный код, который классифицирует объекты по комбинациям цветов.

#### ***Условие***

Необходимо определить, какой из сигналов подает светофор на изображение. Возможных сигналов пять: красный, желтый, зеленый, красный + желтый, никакой — все огни потушены.

Для вас подготовлены несколько файлов «.ру» и набор изображений светофоров. Все файлы находятся в архиве по ссылке: <https://clc.to/HT021-22ATCЗадача1>.

Среди подготовленных файлов есть:

- `helpers.py` — это файл с вспомогательными функциями. Не редактируйте его.
- `eval.py` — файл с ключевыми функциями: предобработки изображений, определения сигнала светофора по изображению. Именно эти две функции вам необходимо дописать!
- `main.py` — это файл проверяет точность работы Вашего алгоритма. Не редактируйте его. `main.py` использует написанные Вами функции из `eval.py` и сверяет истинные метки с предсказанием Вашего алгоритма.

В качестве решения необходимо сдать отредактированный файл `eval.py`.

Видео, которое может помочь при решении задачи: <https://avt.global/cv#trafficlights>.

Короткое видео по работе с платформой Sim: <https://www.youtube.com/watch?v=9ZD52fzCula>.

#### ***Порядок решения:***

1. Ознакомление с данными.

В любой задаче по классификации, а именно ее нам надо решить, сначала необходимо ознакомиться с данными: посмотрите, что представляют собой изображения светофоров.

2. Предварительная обработка.

Все изображения светофоров разные: разного размера, разной яркости, сняты разными камерами. Часто все изображения сначала приводят к одному общему виду, и только после этого передают алгоритму классификации. В файле `eval.py` есть функция для предобработки изображений, измените ее по соб-

ственному усмотрению.

### 3. Алгоритм классификации.

Решите, по каким особенностям изображения можно определить состояние светофора на нем. Эту задачу можно решить без применения методов машинного обучения — не используйте их. В файле `eval.py` есть функция определения сигнала светофора по входному изображению, Вам необходимо ее дописать!

### 4. Проверьте точность своего алгоритма.

Запустите файл `main.py`, он выведет точность работы Вашего алгоритма. Если Вы получите ошибку, разберитесь и устранит ее. Отправляйте решение, только если файл `main` отрабатывает без ошибок!

### 5. Отправьте решение на проверку.

На проверку принимается файл `eval.py`.

Если файл `eval.py` ссылается еще на какие-то файлы, запакуйте их вместе с файлом `eval.py`, в архив «`*.zip`». В архиве должны быть файлы решения, а не одноименная архиву папка.

## **Технические ограничения:**

Пакеты, ориентированные на работу с изображениями и данными, используются на платформе проверки: `matplotlib 2.2.2 numpy 1.14.2 opencv-python 4.2.0.32 pandas 0.22.0 pandocfilters 1.4.2 Pillow 7.0.0 scikit-image 0.15.0 scikit-learn 0.19.1 scipy 1.0.1 sklearn 0.0`.

Используйте совместимые пакеты.

## ***Формат входных данных***

Программа принимает на вход изображение в формате трехмерного массива `numpy`, порядок цветов — `BGR`.

## ***Формат выходных данных***

Программа должна возвращать список из пяти элементов, кодирующих сигнал светофора:

- для красного сигнала — `[1, 0, 0, 0, 0]`;
- для желтого сигнала — `[0, 1, 0, 0, 0]`;
- для зеленого сигнала — `[0, 0, 1, 0, 0]`;
- для красный + желтый сигнала — `[0, 0, 0, 1, 0]`;
- для отсутствующего сигнала — `[0, 0, 0, 0, 1]`.

## ***Решение***

Задача определения сигнала светофора сводится к определению того, какие из трех огней зажжены. Интенсивность свечения объекта хорошо отображает в последней составляющей цвета в цветовом пространстве `HSV`: чем выше значение этой составляющей, тем более светящемуся объекту он принадлежит. Достаточно просуммировать значение пикселей, принадлежащих объекту, и сравнить с порогом, определяемым экспериментально, чтобы решить светится объект или нет. В нашем случае огни светофора на всех фотографиях расположены примерно одинаково, каждый в

своей трети изображения. Для каждой области проводим суммирование и определяем, светится ли она. На основании того, какие огни светофора горят, определяем какой сигнал он подает.

## Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import numpy as np
2 import cv2
3
4 def one_hot_encode(label):
5     # функция преобразования текстового названия сигнала в формат, требуемый для
6     # вывода
7     one_hot_encoded = []
8
9     if label == "red":
10         one_hot_encoded = [1, 0, 0, 0, 0]
11     elif label == "yellow":
12         one_hot_encoded = [0, 1, 0, 0, 0]
13     elif label == "green":
14         one_hot_encoded = [0, 0, 1, 0, 0]
15     elif label == "yellow_red":
16         one_hot_encoded = [0, 0, 0, 1, 0]
17     elif label == "off":
18         one_hot_encoded = [0, 0, 0, 0, 1]
19
20     return one_hot_encoded
21
22 def predict_label(rgb_image):
23     # переводим изображение в цветовое пространство HSV
24     hsv = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2HSV)
25
26     # определяем какие части изображения мы отрежем
27     border_up = 5.6      # отрежем (1/5.6) часть сверху изображения
28     border_down = 9.30    # отрежем (1/9.30) часть сверху изображения
29     border_right = 2.24   # отрежем (1/2.24) часть сверху изображения
30     border_left = 2.22    # отрежем (1/2.22) часть сверху изображения
31
32     # отрезаем от изображения "лишние" части и сохраняем только последний канал
33     # цвета
34     # в последнем канале хранится яркость пикселя
35     hsv = hsv[int(rgb_image.shape[0] / border_up):-int(rgb_image.shape[0] /
36             border_down),
37               int(rgb_image.shape[1] / border_left):-int(rgb_image.shape[1] /
38             border_right), 2]
39
40     # определяем границы трех областей изображения,
41     # в которых располагаются сигнальные огни светофора
42
43     # единицы относительные - высота принята за 1
44     border_red = 0.247      # от 0 до 0.247 - область красного сигнала светофора
45     border_yel_up = 0.244    # от 0.244
46     border_yel_down = 0.778  # до 0.778 - область желтого сигнала светофора
47     border_green = 0.672     # от 0.672 до 1 - область зеленого сигнала светофора
48
49     height, width = hsv.shape # узнаем размер вырезанной части изображения
50     high = hsv[:int(height * border_red)] # вырезаем верхнюю часть - красный
51     # сигнал

```

```

47      # вырезаем среднюю часть - желтый сигнал
48      mid = hsv[int(height * border_yel_up):int(height * border_yel_down)]
49      down = hsv[int(height * border_green):]    # вырезаем нижнюю часть - зеленый
49          ← сигнал
50
51      # вычисляем среднее арифметическое значение пикселей в каждой из областей
52      red_avg = np.average(high)
53      yel_avg = np.average(mid)
54      green_avg = np.average(down)
55
56      # если среднее значение пикселей в области больше чем
57      # в остальных областях на определенное значение
58      # и если среднее значение пикселей в области больше некоторого порога?
59      # то считаем, что сигнал в ней зажжен
60          # если
61      if (red_avg - green_avg > 25.9 and      # красный больше зеленого на 26.9+
62          yel_avg - green_avg > 33.3 and      # желтый больше зеленого на 33.3+
63              red_avg > 197.5 and            # красный выше 197.5
64                  yel_avg > 186.5):        # желтый выше 186.5
65          label = 'yellow_red'           # считаем, что и красный и желтый
65          ← горят - сигнал 'yellow_red'
66
67      elif (red_avg - yel_avg > 4.66 and      # красный больше желтого на 4.6+
68          red_avg - green_avg > 9.12 and      # красный больше зеленого на 9.1+
69              red_avg > 23.17):        # красный выше 23
70          label = 'red'                 # считаем, что красный горит - сигнал 'red'
71
72      elif (yel_avg - red_avg > 4.82 and      # желтый больше красного на 4.8+
73          yel_avg - green_avg > 4.2 and      # желтый больше зеленого на 4.2+
74              yel_avg > 99.96):        # желтый выше 99.9
75          label = 'yellow'               # считаем, что желтый горит - сигнал
75          ← 'yellow'
76
77      elif green_avg > 45.8:    # если зеленый больше 45.8
78          label = 'green'             # считаем, что зеленый горит - сигнал
78          ← 'green'
79      else:
80          label = 'off'                # иначе, все области темные, т.е. ничего не
80          ← горит - сигнал "off"
81
82      encoded_label = one_hot_encode(label)  # переводим текстовое название сигнала в
82          ← формат,
83                                         # необходимый по условию задачи
84
85      return encoded_label

```

### Задача II.1.2. Распознавание дорожных знаков (25 баллов)

Темы: программирование, компьютерное зрение, классификация, нейронная сеть, сверточные слои.

На финале программисту предстоит обучать нейронные сети для классификации объектов на изображении.

## **Условие**

Необходимо определить, какой из дорожных знаков представлен на изображении. Возможных вариантов всего восемь: "no entry", "pedestrian crossing", "road works", "movement prohibition", "parking", "stop", "give way", "artificial roughness".

Для вас подготовлены несколько файлов «.ру» и набор изображений знаков. Все файлы находятся в архиве по ссылке: <https://clc.to/HT021-22ATCЗадача2>.

Среди подготовленных файлов есть:

- `helpers.py` — это файл с вспомогательными функциями, не редактируйте его.
- `eval.py` — файл с ключевыми функциями: предобработки изображений, определения знака на изображении и другими функциями. Вам необходимо внимательно ознакомиться с файлом, решить какие функции требуют изменений и отредактировать их.
- `main.py` — файл, проверяющий точность работы вашего алгоритма. Не редактируйте его. Файл `main.py` использует написанные вами функции из `eval.py` и сверяет истинные метки с предсказанием вашего алгоритма.

В качестве решения необходимо сдать отредактированный файл `eval.py`. Если файл `eval` ссылается на какие-то данные, например, эталонные изображения или файл модели нейронной сети, то их можно добавить к решению. Для этого упакуйте файл `eval.py` и остальные файлы в архив «\*.zip».

Видео, которое может помочь при решении задачи: <https://avt.global/cv#trafficlights>.

Короткое видео по работе с платформой Sim: <https://www.youtube.com/watch?v=9ZD52fzCulA>.

## **Порядок решения**

1. Ознакомление с данными. В любой задаче по классификации, а именно ее нам надо решить, сначала необходимо ознакомиться с данными: посмотрите, что собой представляют изображения знаков.
2. Алгоритм классификации. Решите, по каким особенностям изображения можно определить знак на нем. Эту задачу можно решить без применения методов машинного обучения, однако, нейросети позволяют получить более высокую точность, чем прочие алгоритмы. Используйте тот алгоритм, который Вы способны реализовать. В файле `eval.py` есть функции которые Вам необходимо дописать и исправить!
3. Проверьте точность своего алгоритма. Запустите файл `main.py`, он выведет точность работы Вашего алгоритма. Если Вы получите ошибку, разберитесь и устранит ее. Отправляйте решение, только если файл `main` отрабатывает без ошибок!
4. Отправьте решение на проверку. В качестве решения необходимо сдать файл `eval.py`.

## **Технические ограничения**

Размер решения ограничен: не более 30 МБ. Если алгоритм успешно проверен платформой, то следующее решение можно прислать только через 10 минут. Если алгоритм в ходе проверки выдал сообщение об ошибке, то следующее решения можно прислать сразу.

Пакеты, ориентированные на работу с изображениями и данными, используются на платформе проверки: Python 3.8.10 dlib 19.22.1 imutils 0.5.4 keras 2.7.0 Keras-Preprocessing 1.1.2 matplotlib 3.4.3 numpy 1.21.3 opencv-python 4.5.4.58 pandas 1.3.4 scikit-image 0.18.3 scikit-learn 1.0.1 scipy 1.7.2 tensorflow-cpu 2.7.0.

Используйте совместимые пакеты.

### *Формат входных данных*

Программа принимает на вход изображение в формате трехмерного массива numpy, порядок цветов — BGR.

### *Формат выходных данных*

Программа должна возвращать список из восьми элементов, кодирующих название знака на изображении:

- "no entry" — [1, 0, 0, 0, 0, 0, 0, 0];
- "pedestrian crossing" — [0, 1, 0, 0, 0, 0, 0, 0];
- "road works" — [0, 0, 1, 0, 0, 0, 0, 0];
- "movement prohibition" — [0, 0, 0, 1, 0, 0, 0, 0];
- "parking" — [0, 0, 0, 0, 1, 0, 0, 0];
- "stop" — [0, 0, 0, 0, 0, 1, 0, 0];
- "give way" — [0, 0, 0, 0, 0, 0, 1, 0];
- "artificial roughness" — [0, 0, 0, 0, 0, 0, 0, 1].

### *Решение*

Для решения этой задачи достаточно обучить нейронную сеть — классификатор. Использование сверточной нейронной сети позволит получить наибольшую точность. Для обучения нейронной сети удобно использовать Google Collab и предустановленный tensorflow. Для обучения нейронной сети необходимо подготовить датасет. Датасет состоит из изображений знаков и меток к ним. Метки — это правильные ответы, которые должна дать нейронная сеть. После того, как датасет подготовлен, следует определить архитектуру нейронной сети: набор типов слоев, составляющих сеть и их порядок следования. А также параметры обучения: шаг обучения, оптимизатор, функцию потерь. Как только модель нейронной сети скомпилирована, можно приступить к обучению. Имея файл с обученным нейронным классификатором, остается написать код для его использования.

### *Подготовка датасета*

```

1 import numpy as np
2 import cv2
3 import os
4 import glob
5 import pickle
6
7 def one_hot_encode(label):

```

```

8     """ Функция осуществляет перекодировку текстового "назначения" знака
9     в список элементов, соответствующий формату вывода нейронной сети
10    """
11
12    one_hot_label_dictionary = {"no entry": [1, 0, 0, 0, 0, 0, 0, 0, 0],
13                                "pedestrian crossing": [0, 1, 0, 0, 0, 0, 0, 0, 0],
14                                "road works": [0, 0, 1, 0, 0, 0, 0, 0, 0],
15                                "movement prohibition": [0, 0, 0, 1, 0, 0, 0, 0, 0],
16                                "parking": [0, 0, 0, 0, 1, 0, 0, 0, 0],
17                                "stop": [0, 0, 0, 0, 0, 1, 0, 0, 0],
18                                "give way": [0, 0, 0, 0, 0, 0, 1, 0, 0],
19                                "artificial roughness": [0, 0, 0, 0, 0, 0, 0, 0, 1]}
20
21    one_hot_encoded = one_hot_label_dictionary[label]
22    return one_hot_encoded
23
24 def load_dataset():
25     image_dir = "images/" # путь к папке "images"
26
27     # список с назначениями знаков и подпапок,
28     # по которым разложены изображения разных знаков
29     image_types = ["no entry", "pedestrian crossing", "road works",
30                    "movement prohibition", "parking", "stop",
31                    "give way", "artificial roughness"]
32
33     data = [] # список с изображениями
34     labels = [] # список с метками к изображениям
35
36     # перебираем все подпапки
37     for im_type in image_types:
38         for file in glob.glob(os.path.join(image_dir, im_type, "*")): # перебираем
39             # все файлы в подпапке
40             im = cv2.imread(file) # читаем изображение из файла
41             if not im is None: # если изображение успешно прочитано, то
42                 im = cv2.resize(im, (32, 32)) # уменьшаем изображение до размера
43                 # входа сети
44                 data.append(im) # добавляем изображение в список
45                 labels.append(one_hot_encode(im_type)) # добавляем метку к
46                 # изображению в список
47
48     data = np.array(data, dtype="float") / 255 # делаем из списка массив
49                                         # и нормируем изображения к 1
50     labels = np.array(labels) # делаем из списка массив
51
52     # сохраним обработанные изображения и метки в файлы pickle
53     # сохраняем массив с изображениями
54     with open("/content/drive/MyDrive/NT02021_Preparation/
55               Neural_Net_Not_Binary/data_all_char.pickle", 'wb') as f:
56         pickle.dump(data, f)
57
58     # сохраняем список с лэйблами - метками
59     with open("/content/drive/MyDrive/NT02021_Preparation/
60               Neural_Net_Not_Binary/labels_all_char.pickle", 'wb') as f:
61         pickle.dump(labels, f)

```

## Обучение нейронной сети

```

1 import pickle
2 from sklearn.model_selection import train_test_split

```

```

3  from tensorflow.keras.models import Sequential
4  from tensorflow.keras.layers import Dense # полносвязный слой
5  from tensorflow.keras.layers import Conv2D # сверточный слой
6  from tensorflow.keras.layers import MaxPooling2D # слой подвыборки
7
8  # слой, который приводит данные к виду одномерному вектору
9  from tensorflow.keras.layers import Flatten # слой, который приводит данные к
   ↵ виду
10 from tensorflow.keras.layers import Activation # слой функций активации
11 from tensorflow.keras.layers import BatchNormalization # пакетная нормализация
12 from tensorflow.keras.optimizers import Adam
13 from tensorflow.keras.callbacks import ModelCheckpoint
14
15 # загружаем массив с изображениями
16 with open("sign_image.pickle", 'wb') as f:
17     data = pickle.load(f)
18 print("Data loaded")
19
20 # загружаем массив с лейблами-метками
21 with open("sign_label.pickle", 'wb') as f:
22     labels = pickle.load(f)
23 print("Labels loaded")
24
25 # делим подготовленные данные на тренировочную и валидационную часть
26 (trainX, testX, trainY, testY) = train_test_split(data, labels,
27                                                 test_size=0.15,
28                                                 random_state=42)
29
30 # определяем архитектуру нейронной сети
31 model = Sequential() # модель последовательная
32 inputShape = (32, 32, 3) # форма входных данных
33 chanDim = -1 # последняя ось входного массива - отвечает за цвет
   ↵ пикселя
34
35 # добавляем слои к нейронной сети
36 model.add(Conv2D(8, (5, 5), padding="same", input_shape=inputShape))
37 model.add(Activation("relu"))
38 model.add(BatchNormalization(axis=chanDim))
39 model.add(MaxPooling2D(pool_size=(2, 2)))
40
41 model.add(Conv2D(16, (3, 3), padding="same"))
42 model.add(Activation("relu"))
43 model.add(BatchNormalization(axis=chanDim))
44 model.add(Conv2D(16, (3, 3), padding="same"))
45 model.add(Activation("relu"))
46 model.add(BatchNormalization(axis=chanDim))
47 model.add(MaxPooling2D(pool_size=(2, 2)))
48
49 model.add(Conv2D(32, (3, 3), padding="same"))
50 model.add(Activation("relu"))
51 model.add(BatchNormalization(axis=chanDim))
52 model.add(Conv2D(32, (3, 3), padding="same"))
53 model.add(Activation("relu"))
54 model.add(BatchNormalization(axis=chanDim))
55 model.add(MaxPooling2D(pool_size=(2, 2)))
56
57 model.add(Flatten())
58 model.add(Dense(128))
59 model.add(Activation("relu"))
60 model.add(BatchNormalization())

```

```

61
62     model.add(Dense(8))
63     model.add(Activation("softmax"))
64
65     # определяем оптимизатор для обучения и его параметры
66     opt = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False)
67
68     # компилируем модель нейронной сети
69     # функция потерь - категориальная кроссэнтропия
70     model.compile(loss="categorical_crossentropy",
71                     optimizer=opt,
72                     metrics=["accuracy"]) # метрика - точность
73
74     # Определяем функцию, сохраняющую модель с наибольшей точностью
75     checkpointer = ModelCheckpoint(filepath='Model_Net.h5', # Имя файла с лучшей
76                                    → моделью
77                                    verbose=1, save_best_only=True)
78
79     # определяем параметры обучения и запускаем его
80     H = model.fit(trainX, trainY, validation_data=(testX, testY),
81                   epochs=7, batch_size=32,
82                   shuffle=True,
83                   callbacks=[checkpointer])

```

### Код для проверки

```

1  import cv2
2  import numpy as np
3  from tensorflow import keras
4
5  def get_standatr_signs():
6      return []
7
8  def load_final_model():
9      """ Функция осуществляет загрузку модели нейронной сети из файла. """
10     model = keras.models.load_model("Model_Net.h5")
11     return model
12
13 def one_hot_encode(label):
14
15     """ Функция осуществляет перекодировку текстового "названия" сигнала
16     в список элементов, соответствующий выходному сигналу
17     """
18
19     one_hot_label_dictionary = {"no entry": [1, 0, 0, 0, 0, 0, 0, 0],
20                                "pedestrian crossing": [0, 1, 0, 0, 0, 0, 0, 0],
21                                "road works": [0, 0, 1, 0, 0, 0, 0, 0],
22                                "movement prohibition": [0, 0, 0, 1, 0, 0, 0, 0],
23                                "parking": [0, 0, 0, 0, 1, 0, 0, 0],
24                                "stop": [0, 0, 0, 0, 0, 1, 0, 0],
25                                "give way": [0, 0, 0, 0, 0, 0, 1, 0],
26                                "artificial roughness": [0, 0, 0, 0, 0, 0, 0, 1]}
27
28     one_hot_encoded = one_hot_label_dictionary[label]
29     return one_hot_encoded
30
31 def standardize_input(image):
32     # изменяем размер до размера, требуемого на входе нейронной сети
33     image = cv2.resize(image, (32, 32))

```

```

34     standard_im = np.array(image, dtype="float") / 255 # нормируем изображение к 1
35
36     return standard_im
37
38 def predict_label(image, model, standart_signs):
39
40     standard_im = standardize_input(image) # приводим изображение к стандартному
41     # виду
42     predict = model.predict(standard_im)[0] # получаем ответ от нейронной сети
43     output = [0, 0, 0, 0, 0, 0, 0, 0] # заготовка для выходной метки
44
45     # ставим 1 в том месте, где нейронная сеть наиболее уверена
46     output[np.argmax(output)] = 1
47
48     return output

```

## Командная часть

### **Задача II.2.1. Детектирование светофоров (20 баллов)**

Темы: программирование, компьютерное зрение, детектирование, нейронные сети, нейросетевые детекторы.

На финале программисту предстоит разрабатывать программный код, который детектирует объекты на изображении с камеры.

#### **Условие**

Необходимо определить, где на изображении расположен светофор, и какой из сигналов он подает. Возможных сигналов пять: красный, желтый, зеленый, красный + желтый, никакой — все огни потушены.

Для Вас подготовлены несколько файлов «.ру», набор изображений светофоров и аннотации к нему. Все файлы находятся в архиве: <https://clc.to/HT0-21-22-ATC-Задача3>.

Среди подготовленных файлов есть:

- **eval.py** — файл с ключевыми функциями: предобработки изображений, загрузки модели машинного обучения, детектирования светофора и распознавания его сигнала. Именно эти функции Вам необходимо дописать!
- **main.py** — файл проверяющий точность работы Вашего алгоритма. Не редактируйте его. **main.py** использует, написанные Вами функции из **eval.py** и сверяет истинные метки с предсказанием Вашего алгоритма.
- **annotations.csv** — файл устанавливающий соотношение между изображениями, координатами светофоров и их сигналами. В каждой строке файла содержится путь к файлу с изображением, название сигнала, который подает светофор и координаты ограничивающей светофор рамки.

В качестве решения, необходимо сдать отредактированный файл **eval.py**.

#### **Порядок решения:**

1. Ознакомление с данными.

В любой задаче по детектированию и классификации, а именно их нам надо

решить, сначала необходимо ознакомиться с данными: посмотрите, что из себя представляют размеченные изображения с светофорами.

## 2. Разметка собственного датасета.

Чтобы решить поставленную задачу, Вам необходимо разметить собственный датасет. Скачайте подготовленные для Вас видео: <https://clc.to/HT0-21-22-ATC-Видео>.

Пример размеченных изображений у Вас уже есть.

## 3. Алгоритм классификации.

Решите, используя какой алгоритм детектирования Вы собираетесь применять. Вы будете использовать только алгоритм детектирования или будете использовать после него алгоритм классификации? В файле `eval.py` есть функция определения сигнала светофора и координат ограничивающей его рамки, Вам необходимо ее дописать!

## 4. Предварительная обработка.

Изображения, прежде чем передать алгоритмам детектирования или распознавания, необходимо привести к одному общему виду. В файле `eval.py` есть функция для предобработки изображений, измените ее по собственному усмотрению.

## 5. Проверьте точность своего алгоритма.

Запустите файл `main.py`, он выведет точность работы Вашего алгоритма. Если Вы получите ошибку, разберитесь и устраните ее. Отправляйте решение, только если файл `main` отрабатывает без ошибок!

## 6. Отправьте решение на проверку.

Нажмите кнопку «Прислать решение» и выберите файл `eval.py`.

### **Технические ограничения:**

Размер решения ограничен: не более 30 МБ.

Пакеты, ориентированные на работу с изображениями и данными, использующиеся на платформе проверки: Python 3.8.10 dlib 19.22.1 imutils 0.5.4 keras 2.7.0 Keras-Preprocessing 1.1.2 matplotlib 3.4.3 numpy 1.21.3 opencv-python 4.5.4.58 pandas 1.3.4 scikit-image 0.18.3 scikit-learn 1.0.1 scipy 1.7.2 tensorflow-cpu 2.7.0.

Используйте совместимые пакеты.

### **Формат входных данных**

Программа принимает на вход изображение в формате трехмерного массива `numpy`, порядок цветов — BGR.

### **Формат выходных данных**

Программа должна возвращать список списков с текстовыми метками объектов и координатами, ограничивающими их рамки: `[[label, (x, y, x2, y2)]]`, где

`label` — текстовая метка с названием сигнала светофора,

`x` и `y` — координаты левого верхнего угла ограничивающей рамки,

`x2` и `y2` — координаты правого нижнего угла ограничивающей рамки.

## Примеры вывода

Если обнаружили два светофора:

```
[["off (10, 14, 18, 25)", "yellow+red (110, 114, 118, 215)"]]
```

Если обнаружили один светофор:

```
[["green (10, 11, 18, 15)"]]
```

Если если не обнаружили светофоры: []

## Решение

Для решения этой задачи достаточно обучить нейронную сеть — детектор. Детектор YOLO 4, позволяет получить наибольшую точность. Для обучения нейронной сети удобно использовать Google Collab и фреймворк `darknet`. Фреймворк необходимо установить в Google Collab, следую инструкциям из официального репозитория: <https://github.com/ntomaterials/darknet>.

После того, как `darknet` установлен, необходимо разметить датасет для обучения детектора. В качестве инструмента разметки может использоваться `LabelImg`: <https://github.com/ntomaterials/labelImg>.

Обучение нейронной сети выполняется с помощью команд фреймворка. В результате обучения должны получиться два файла: файл с конфигурацией сети и файл с весовыми коэффициентами, «`yolov4.cfg`» и «`yolov4.weights`», соответственно. Остается написать код для применения детектора к изображениям не из датасета.

## Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import cv2
2
3 def load_final_model():
4     # загружаем конфигурацию и весовые коэффициенты из файлов
5     net = cv2.dnn.readNet("yolov4.weights", "yolov4.cfg")
6     net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV) # выбираем бэкэнд
7     # указываем в качестве вычислительного устройства CPU
8     net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)
9
10    # формируем модель детектора, для дальнейшего использования
11    model = cv2.dnn_DetectionModel(net)
12
13    model.setInputParams(size=(512, 512), # размер изображения
14                          scale=1 / 255, # делим значения всех пикселей на 255
15                          swapRB=True) # меняем местами каналы цвета B и R
16
17    return [model] # возвращаем список моделей
18
19
20 def standardize_input(image):
21     return image
22
23
24 def predict_label_and_box(image, model):
25     standard_im = standardize_input(image)
```

```

26
27      # получаем результаты детектирования от нейронной сети
28      # standard_im - изображение, на котором мы ищем объекты
29      # 0.3 - порог уверенности в детектировании, все что ниже будет отброшено
30      # 0.4 - прог NMS, рамки с IoU (для наиболее вероятной рамки) ниже будут удалены
31      classes, scores, boxes = model.detect(standard_im, 0.3, 0.4)
32
33      # В этом список будем добавлять ограничивающие рамки, которые нашел детектор
34      predicted_label_and_box = []
35      # Список имен классов
36      class_names = ['green', 'off', 'red', 'yellow', 'yellow+red']
37
38      # перебираем все найденные объекты, нас интересуют номер класса и
39      # → ограничивающая рамка
40      for (classid, score, box) in zip(classes, scores, boxes):
41          # добавляем текстовую метку класса и координаты ограничивающей рамки
42          predicted_label_and_box.append([class_names[classid],
43                                           (box[0], box[1], box[0]+box[2],
44                                           → box[1]+box[3])])
44
44      return predicted_label_and_box

```

### **Задача II.2.2. Полет квадрокоптера (20 баллов)**

Темы: программирование, Robot Operating System (ROS), летный алгоритм, полетное задание.

На финале программисту предстоит писать программный код для квадрокоптера.

#### **Условие**

Необходимо создать полетное задание, в котором квадрокоптер взлетает из заданной точки «дом», пролетает по всем точкам из массива `coordinates`, возвращается на точку «дом» и совершает посадку.

Для Вас подготовлен файл `eval.py`. Все файлы находятся в архиве: <https://clc.to/HT0-21-22-ATC-Задача4>.

Он содержит шаблон полетного задания. Вам необходимо внимательно ознакомиться с файлом, решить какие части кода требуют изменений и отредактировать их. В качестве решения необходимо сдать на проверку отредактированный файл `eval.py`.

Внимание! Для решения задачи Вы вносите изменения только в файл `eval.py`. Никаких дополнительных библиотек, кроме подключенных в коде `eval.py`, Вы использовать и устанавливать не должны.

#### **Порядок решения:**

1. Ознакомление с исходными данными.

Полетное задание представляет собой массив точек, по которым последовательно летит квадрокоптер. Исходные данные — точка «дом» (переменная `home_point`) и массив с координатами точек (`coordinates`) — изменению не подлежат.

2. Поиск и исправление ошибки в имеющемся коде.

Вам предлагается дописать код алгоритма для осуществления возврата на точ-

ку «дом» и посадки в блоке с комментарием «# дописать здесь».

Алгоритм полета по точкам уже реализован и работает верно. Однако внимательно изучите исходные условия, при которых коптер осуществляет взлет. В алгоритм вкрадалась ошибка, которую Вы должны обнаружить и исправить.

### 3. Отправьте решение на проверку.

Нажмите кнопку «Прислать решение» и выберите файл `eval.py`. Если Вам удалось исправить ошибки в алгоритме и квадрокоптер пролетел по маршруту, то Вы получите результат: 1.0.

## *Решение*

Для решения этой задачи достаточно обучить нейронную сеть — детектор. Детектор YOLO 4, позволяет получить наибольшую точность. Для обучения нейронной сети удобно использовать Google Collab и фреймворк `darknet`. Фреймворк необходимо установить в Google Collab, следуя инструкциям из официального репозитория: <https://github.com/ntomaterials/darknet>.

После того, как `darknet` установлен, необходимо разметить датасет для обучения детектора. В качестве инструмента разметки может использоваться LabelImg: <https://github.com/ntomaterials/labelImg>.

Обучение нейронной сети выполняется с помощью команд фреймворка. В результате обучения должны получиться два файла: файл с конфигурацией сети и файл с весовыми коэффициентами, «`yolov4.cfg`» и «`yolov4.weights`», соответственно. Остается написать код для применения детектора к изображениям не из датасета.

## *Пример программы-решения*

Ниже представлено решение на языке Python 3.

```

1 import rospy
2 from gs_flight import FlightController, CallbackEvent
3 from gs_board import BoardManager
4
5 rospy.init_node("flight_test_node") # инициализируем ноду в системе ROS
6 home_point = [2, 2, 1] # задаем точку «дом»
7 coordinates = [[0, 1, 1],
8                 [1, 1, 1],
9                 [1, 0, 1],
10                [0, 0, 1]] # создаем массив точек, по которым будет лететь
11               → квадрокоптер
12 run = True # переменная отвечающая за работу программы, значение True программа
13               → выполняется
14 position_number = 0 # счетчик пройденных точек
15
16 def callback(event): # функция обработки событий Автопилота
17     global ap
18     global run
19     global coordinates
20     global position_number
21
22     event = event.data # преобразуем сообщение ROS std_msgs/Int32 в Питоновский int
23     # если событие Автопилота ENGINES_STARTED( моторы заведены), то выполняем взлет

```

```

24     # иначе делаем проверку на другое событие Автопилота
25     if event == CallbackEvent.ENGINES_STARTED:
26         print("engine started") # выводим на экран сообщение, что двигатели
27             ↪ заведены
28         ap.takeoff() # отдаём команду на взлёт, как только она будет выполнена
29             #Автопилот сгенерирует событие TAKEOFF_COMPLETE (взлёт завершен)
30         # если событие Автопилота TAKEOFF_COMPLETE (взлёт завершен)
31         # то обнуляем счетчик точек и отдаём команду Автопилоту переместится в нулевую
32             ↪ точку из массива,
33         # иначе делаем проверку на другое событие Автопилота
34     elif event == CallbackEvent.TAKEOFF_COMPLETE:
35         print("takeoff complete") # выводим на экран сообщение, что взлёт закончен
36         position_number = 0 # обнуляем счетчик точек
37         # отдаём Автопилоту команду на перемещение в нулевую точку,
38         # как только команда завершится
39         # Автопилот сгенерирует POINT_REACHED(точка достигнута)
40         ap.goToLocalPoint(coordinates[position_number][0],
41                             coordinates[position_number][1],
42                             coordinates[position_number][2])
43         # если событие Автопилота POINT_REACHED (точка достигнута),
44         # то отдаём команду на перемещение в следующую точку или
45         # отдаём команду на приземление
46     elif event == CallbackEvent.POINT_REACHED:
47         print("point {} reached".format(position_number)) # выводим на экран номер
48             ↪ достигнутой точки
49         position_number += 1 # наращиваем счетчик пройденных точек на 1
50         # делаем проверку: если количество пройденных точек меньше чем количество
51             ↪ точек
52         # в массиве со всеми точками, то выполняем перемещение в следующую точку,
53         # иначе делаем проверку: если количество пройденных точек равно количеству
54             ↪ точек
55         # в массиве со всеми точками, то выполняем перемещение в точку «Дом»,
56         # иначе выполняем посадку
57     if position_number < len(coordinates):
58         # перемещение в следующую точку
59         # как только команда завершится
60         # Автопилот сгенерирует POINT_REACHED(точка достигнута)
61         ap.goToLocalPoint(coordinates[position_number][0],
62                             coordinates[position_number][1],
63                             coordinates[position_number][2])
64         # перемещение в точку «Дом»
65         # как только команда завершится
66         # Автопилот сгенерирует POINT_REACHED(точка достигнута)
67         ap.goToLocalPoint(home_point[0], home_point[1], home_point[2])
68     else:
69         ap.landing()
70         # если событие Автопилота равно COPTER_LANDED (совершена посадка),
71         # то прекращаем выполнение программы
72     elif event == CallbackEvent.COPTER_LANDED:
73         print("finish programm")
74         # устанавливаем переменную,
75         # отвечающую за работу программы в значение False - программа должна
76             ↪ прекратиться
77         run = False
78
79     # создаем объект класса получения бортовой информации
80     board = BoardManager()
81     # создаем объект управления полета Автопилотом, передаем функцию обработки
82             ↪ событий,

```

```

78 # как только придет событие от Автопилота эта функция будет вызвана
79 ap = FlightController(callback)
80
81 # переменная, отвечающая за однократное выполнения блока предстартовой подготовки.
82 once = False
83
84 # цикл во время выполнение которого работает функция callback
85 # Этот цикл работает пока переменная тип в значении True и пока ROS включен
86 while not rospy.is_shutdown() and run:
87     # если плата Автопилота подключена к Raspberry и переменная once равна False,
88     # то выполняем предстартовую подготовку
89     if board.runStatus() and not once:
90         print("start programm")
91         # отдает команду Автопилоту выполнить предстартовую подготовку,
92         # как только она будет выполнена
93         # Автопилот сгенерирует событие ENGINES_STARTED (Двигатели заведены)
94         ap.preflight()
95         # устанавливаем в True, чтобы больше не выполнять блок предстартовой
96         # подготовки
97         once = True
98     pass

```

### **Задача II.2.3. Распознавание маркировок (20 баллов)**

Темы: программирование, компьютерное зрение, контурный анализ, аффинные преобразования, пространственные преобразования изображений.

На финале программисту предстоит разрабатывать программный код, который распознает цветовые маркировки грузов и на их основании производит сортировку грузов.

#### **Условие**

Необходимо декодировать цветную маркировку на изображении.

Цветовая маркировка представляет собой 16 цветных квадратов. Квадраты образуют матрицу  $4 \times 4$ . Цвет квадрата кодирует цифру.

```

color_dict = {0: (255, 255, 255), # white
              1: (0, 0, 0), # black
              2: (0, 0, 255), # red
              3: (0, 136, 255), # orange
              4: (0, 255, 255), # yellow
              5: (0, 255, 0), # green
              6: (255, 255, 0), # cyan
              7: (255, 0, 0), # blue
              8: (255, 0, 136), # violet
              9: (255, 0, 255)} # magenta
# bgr

```

Таким образом, цветовая маркировка может быть расшифрована как последовательность из 16 цифр. Получить верную последовательность цифр можно, только если читать маркировку слева направо и сверху вниз.

Представьте, что маркировка нанесена на стенку коробки. Коробка перевернулась, Вы все еще видите маркировку, но не знаете где у нее верхний левый угол, с

которого надо начинать ее чтение. Чтобы всегда можно было найти квадрат, с которого надо начинать чтение маркировки, введено следующие правила. Квадраты соседние по вертикали и горизонтали для левого верхнего квадрата в сумме всегда дают 7. А соседи левого нижнего квадрата никогда не дают в сумме 7. Таким образом, анализируя цвета соседей угловых квадратов, Вы всегда можете определить, откуда надо начинать читать маркировку.

Для Вас подготовлены несколько файлов «.ру» и набор изображений цветных маркировок. Все файлы находятся в архиве: <https://clc.to/HTO-21-22-ATC-Задача5>.

Среди подготовленных файлов есть:

- `eval.py` — файл с ключевыми функциями: предобработки изображений, загрузки модели машинного обучения, декодирования маркировки. Именно эти функции Вам необходимо дописать!
- `main.py` — файл проверяющий точность работы Вашего алгоритма. Не редактируйте его. `main.py` использует, написанные Вами функции из `eval.py` и сверяет истинные расшифровки маркировок с теми, которые дает Ваш алгоритм.
- `annotations.csv` — файл устанавливающий соотношение между изображениями цветных маркировок и их расшифровками. В каждой строке файла содержится путь к файлу с изображением и его расшифровка.

В качестве решения, необходимо сдать отредактированный файл `eval.py`.

#### **Порядок решения:**

0. Еще раз прочтайте условие задачи, оно не простое.

1. Ознакомление с исходными данными.

В любой задаче по детектированию и классификации, а именно их нам надо решить, сначала необходимо ознакомиться с данными: посмотрите, что собой представляют изображения экземпляров цветовой маркировки.

2. Какой алгоритм использовать?

Эту задачу можно решить без использования алгоритмов машинного обучения, но Вы можете их использовать. Подумайте, какие особенности изображения Вы можете использовать для декодирования маркировки. Определитесь какие алгоритмы Вы будете использовать. В файле `eval.py` есть функция декодирования маркировки, Вам необходимо ее дописать!

3. Проверьте точность своего алгоритма.

Запустите файл `main.py`, он выведет точность работы Вашего алгоритма. Если Вы получите ошибку, разберитесь и устраните ее. Отправляйте решение, только если файл `main` отрабатывает без ошибок!

4. Отправьте решение на проверку.

Нажмите кнопку «Прислать решение» и выберите файл `eval.py`.

#### **Технические ограничения**

Размер решения ограничен: не более 30 МБ.

Пакеты, ориентированные на работу с изображениями и данными, использующиеся на платформе проверки: Python 3.8.10 dlib 19.22.1 imutils 0.5.4 keras 2.7.0 Keras-Preprocessing 1.1.2 matplotlib 3.4.3 numpy 1.21.3 opencv-python 4.5.4.58 pandas 1.3.4 scikit-image 0.18.3 scikit-learn 1.0.1 scipy 1.7.2 tensorflow-cpu 2.7.0

Используйте совместимые пакеты.

## *Формат входных данных*

Программа принимает на вход изображение в формате трехмерного массива `numpy`, порядок цветов — BGR.

## *Формат выходных данных*

Текстовая строка, состоящая из 16 цифр.

Примеры вывода:

"1779017760567636"

"0620121015168086"

## *Решение*

Решение задачи можно разбить на несколько этапов: восстановление квадратной формы цветовой маркировки, замена цветов цифрами, определение первой цифры в маркировке.

Для восстановления формы маркировки необходимо воспользоваться контурным анализом и получить контур четырехугольника маркировки. Затем необходимо найти вершины контура и их координаты. Зная их, можно воспользоваться пространственным преобразованием и трансформировать четырехугольник маркировки в квадрат.

Теперь не составит труда заменить цвета на цифры. Квадраты, кодирующие цифры, имеют одинаковый размер. Значит, мы легко можем вычислить координаты центра каждого квадрата и сравнить цвет центрального пикселя с цветами из словаря. Нам необходимо найти ближайший цвет из словаря. Это такой цвет, у которого сумма разностей по каждому каналу цвета с нашим пикселем минимальная.

Заменив все цвета на цифры, мы получаем матрицу  $4 \times 4$ . Теперь необходимо определить, с какого угла ее правильно читать. Поскольку возможных вариантов ориентации матрицы всего 4, проверяем соблюдается ли ключевое условие для текущей ориентации, если нет — поворачиваем матрицу на 90 градусов. Если условие соблюдается, то превращаем матрицу в строку цифр, читая матрицу слева направо, сверху вниз.

## *Пример программы-решения*

Ниже представлено решение на языке Python 3.

```

1 # -*- coding: utf-8 -*-
2 import cv2
3 import numpy as np
4
5 def load_final_model():
6     model = []
7     return model
8
9 def standardize_input(image):
10    standard_im = image
11    return standard_im

```

```

12
13
14 def predict_sequence_of_digital(image, model):
15     # создаем словарь соответствия цифр и цветов, кодирующих цифры
16     color_dict = {0: (255, 255, 255), # white
17                   1: (0, 0, 0), # black
18                   2: (0, 0, 255), # red
19                   3: (0, 136, 255), # orange
20                   4: (0, 255, 255), # yellow
21                   5: (0, 255, 0), # green
22                   6: (255, 255, 0), # cyan
23                   7: (255, 0, 0), # blue
24                   8: (255, 0, 136), # violet
25                   9: (255, 0, 255)} # magenta
26
27     # бинаризуем изображение так, чтобы белым остался только серый фон
28     # и после этого, инвертируем цвета изображения, т.е. белой будет вся цветовая
29     # маркировка
30     gray = 255 - cv2.inRange(image, (152, 152, 152), (168, 168, 168))
31
32     # находим контуры на изображении, иерархию - не используем
33     cnts = cv2.findContours(gray, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
34
35     # выбираем контур с максимальной площадью
36     cnt = max(cnts, key=cv2.contourArea)
37
38     # уменьшаем количество вершин в контуре, в идеале мы получим квадрат
39     cnt = cv2.approxPolyDP(cnt, 0.015 * cv2.arcLength(cnt, True), True)
40
41     # Мы собираемся преобразовать растянутую метку в квадрат
42     # Создаем массив с координатами вершин этого квадрата
43     out = np.array([[0, 0], [127, 0], [127, 127], [0, 127]])
44
45     # приводим форму массива с контуром к форме, созданного ранее массива
46     # такая форма нужна для функции getPerspectiveTransform()
47     cnt = cnt.reshape(out.shape)
48
49     # вычисляем матрицу преобразования, для преобразования исходной метки в квадрат
50     M = cv2.getPerspectiveTransform(cnt.astype(np.float32), out.astype(np.float32))
51
52     # преобразуем метку в квадрат, в соответствии с вычисленной ранее матрицей
53     # и сохраняем на изображении 128 на 128 пикселей
54     warped = cv2.warpPerspective(image, M, (128, 128))
55
56     # создаем массив 4 на 4 с 0, будем записывать в него расшифровку маркировки
57     mat = np.zeros((4, 4), dtype=np.uint8)
58
59     # перебираем 16 точек внутри квадрата маркировки
60     # каждая точка соответствует центру квадра, кодирующего цифру
61     for x in range(4):
62         for y in range(4):
63             p = warped[16 + y * 32, 16 + x * 32] # сохраняем цвет пикселя в центре
64             # квадрата
65
66             # сравниваем цвет пикселя с цветами из словаря, находим ближайший
67             # записываем число, которое кодирует цвет в массив mat
68             mat[y, x] = min(color_dict.keys(), key=lambda k: sum(abs(p[i] -
69             color_dict[k][i]) for i in range(3)))
70
71     # Теперь, наша задача определить с какого угла нужно начинать чтение маркировки

```

---

```
69      # всего четыре возможных варианта ориентации метки
70      for i in range(4):
71          # Если сумма соседей левого верхнего числа равна семи,
72          # а сумма соседей левого нижнего не равна 7,
73          # то считаем, что метка правильно ориентирована и выходим из цикла
74          if (mat[0, 1] + mat[1, 0]) == 7 and (mat[-1, 1] + mat[-2, 0]) != 7:
75              break
76          else:
77              # Иначе, поворачиваем метку на 90 градусов
78              mat = np.rot90(mat)
79
80      # преобразуем массив с цифрами в текстовую строку
81      sequence_of_digital = '' # будем дописывать к ней цифры
82      for row in mat: # перебираем строки
83          for digit in row: # перебираем цифры в строках
84              sequence_of_digital += str(digit) # дописываем цифру в конец строки
85      return sequence_of_digital
```

# Заключительный этап

## Индивидуальный предметный тур

### Информатика. 8–11 класс

#### *Задача III.1.1.1. Беспилотная автоколонна (8 баллов)*

Одна из уже реализованных концепций беспилотных автомобилей заключается в создании автоколонн, лидером которых управляет, например, человек-водитель, а все остальные автомобили в этой колонне следуют под управлением автопилота, ориентируясь на показания впереди идущего транспорта. Как правило, таким образом выстраиваются грузовые автоколонны.

Допустим, в стартовый момент времени из пункта  $A$  стартовал лидер колонны. Далее происходили события двух типов:

« $+x$ » — в конец колонны добавился автомобиль с грузом  $x$  тонн.

« $-n$ » — последние  $n$  автомобилей колонны достигли своей цели, отсоединились от автоколонны и съехали на разгрузку.

По списку указанных событий для каждого запроса второго вида « $-n$ » вывести суммарное количество груза, которое доставили эти  $n$  грузовиков в пункт своего назначения.

#### *Формат входных данных*

В первой строке содержится число  $m$  — общее количество событий,  $2 \leq m \leq 10^5$ . Далее в  $m$  следующих строках содержатся описания событий.

Событие вида « $+x$ » означает, что в конец автоколонны добавляется еще один грузовик с грузом  $x$  тонн, при этом  $x$  целое и  $1 \leq x \leq 10^9$ .

Событие вида « $-n$ » означает, что последние  $n$  автомобилей колонны достигли пункта назначения и отсоединились от колонны. Гарантируется, что число  $n$  положительное целое и не превосходит числа грузовых автомобилей без лидера в колонне на момент такого события. В списке событий есть, по крайней мере, одно событие второго типа.

#### *Формат выходных данных*

Для каждого события второго вида « $-n$ » вывести суммарное количество груза, которое доставили эти  $n$  грузовиков в пункт своего назначения в отдельной строке.

## Примеры

### Пример №1

Стандартный ввод	Стандартный вывод
<pre>15 +2 +7 +4 -1 +10 +1 +4 -2 +5 +5 +5 +2 -5 +2 -3</pre>	<pre>4 5 27 11</pre>

## Решение

При решении задачи нужно промоделировать состояние автоколонны, добавляя в конец и удаляя из конца грузовики. Весь описанный в задаче процесс явно похож на работу с такой структурой, как стек. Действительно, грузовики добавляются и удаляются из конца колонны по принципу «последний пришел — первый ушел», а это и есть протокол работы со стеком. Стек можно как промоделировать при помощи таких структур, как вектор в C++ или `list` в Python, так и воспользоваться встроенными контейнерами для стека. В любом случае каждый грузовик добавляем в конец как число, равное его грузоподъемности для каждого события вида  $+x$ , и для каждого события вида  $-n$  удаляем из конца  $n$  чисел, суммируя их и выдавая эту сумму в ответ.

### Задача III.1.1.2. Передовые кондитерские технологии (10 баллов)

Сегодня можно печатать даже торты! Предлагаем оценить скорость доработки торта «Графские развалины». Сама основа торта, собственно «графские развалины» уже изготовлена, осталось оформить ее кремом. Торт изначально не содержит крема и схематически может быть представлен как набор из  $n$  «столбиков», каждый из которых имеет высоту  $h_i$  ячеек.

Крем будет наносить не кондитер, а специальный принтер. Особенностью работы этого принтера является его удвоенная скорость работы, которая достигается

за счет двух рабочих головок, работающих параллельно. За одну операцию принтер заполняет кремом ровно две произвольные ячейки рабочего пространства. При необходимости принтер может создавать новый столбик, заполненный исключительно кремом.

Торт будет считаться готовым, если все составляющие его столбики будут иметь одинаковую высоту. В каждый столбик можно добавлять сколько угодно ячеек с кремом.

Требуется по описанию исходных «графских развалин» определить минимальное количество операций, после которых торт будет считаться готовым.

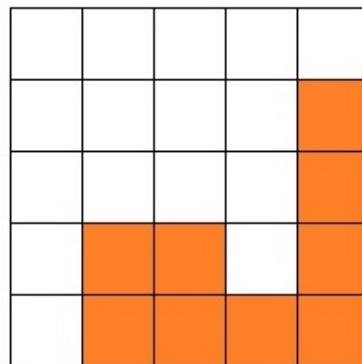
### ***Формат входных данных***

В первой строке содержится одно целое число  $n$  — количество столбиков в исходной заготовке,  $1 \leq n \leq 10^5$ .

В следующей строке содержатся  $n$  целых чисел  $h_i$  через пробел — высоты каждого столбика,  $1 \leq h_i \leq 10^9$ .

### ***Формат выходных данных***

Вывести одно число — количество операций, после которых столбцы торта будут иметь одинаковую высоту.



### ***Примеры***

#### *Пример №1*

<b>Стандартный ввод</b>
4
2 2 1 4
<b>Стандартный вывод</b>
8

*Пример №2*

<b>Стандартный ввод</b>
4
3 3 1 3
<b>Стандартный вывод</b>
1

*Комментарий к примеру 1*

На рисунке оранжевым указаны исходные столбцы, белым — добавленные ячейки с кремом. Так как на каждом шагу будет добавляться по две новые ячейки, то понадобится добавить один столбец и одну строку, чтобы торт стал иметь одинаковую по всем столбцам высоту. Минимальное количество операций тогда будет равно 8.

*Решение*

Очевидно, что минимальная высота полученного торта не может быть меньше максимального исходного столбика. Пусть  $mx$  — высота наибольшего столбика, а  $n$  — число всех столбиков. Вычислим величину  $d = n \cdot mx - sum$ , где  $sum$  равно сумме высот всех столбиков. Эта разность равна минимально возможному количеству ячеек, которые нужно заполнить кремом. Если она четна, то ответ равен  $d/2$ . Если нечетна, то нужно либо добавить еще один слой сверху, либо еще один столбец из крема сбоку, либо и то и другое одновременно. Для тех случаев, когда добавка нечетна (а хотя бы в одном из этих трех случаев это будет так, а значит ее сумма с числом  $d$  станет четной), нужно выбрать тот случай, когда она минимально возможная и вывести половину от суммарной добавки.

*Задача III.1.1.3. Доставка комплектующих (22 баллов)*

При проектировании производства один из важных вопросов, который нужно решить, связан с организацией доставки необходимых комплектующих в точки сборки. Раньше для этого использовали конвейер, но теперь, когда производство из линейного процесса превратилось в разветвленный орграф, удобнее организовать его на плоскости, а для доставки комплектующих в нужные точки производства использовать автономные платформы.

Пусть производственное помещение имеет форму прямоугольника из  $n$  строк и  $m$  столбцов и разбито на  $n \times m$  ячеек. Платформа имеет автономное управление и для ориентации использует оптическую систему позиционирования. Платформа имеет квадратную форму, и может перемещаться в любом из четырех направлений на смежную по стороне ячейку. Для того, чтобы платформа могла переместиться в каком-либо из направлений и не потерять ориентацию, либо перед ней, либо после нее **по линии движения** на расстоянии не более чем  $k$  должна быть светоотражающая конструкция. При перемещении платформа включает передний и задний относительно движения датчик позиционирования и хотя бы один из этих двух датчиков должен фиксировать в любой момент перемещения светоотражающую конструкцию на расстоянии не более, чем  $k$ . Боковые — относительно текущего направления движения — датчики находятся в выключенном состоянии. Обращаем внимание на то,

---

что в процессе текущего перемещения, конструкция, зафиксированная для позиционирования, не может быть изменена на другую. Переключение между точками позиционирования возможно только между перемещениями.

Производственное помещение по периметру огорожено такими светоотражающими конструкциями, кроме того, некоторые внутренние ячейки также заняты светоотражающими конструкциями. Перемещаться в ячейку с такой конструкцией платформа не может.

На данный момент производится планирование расстановки этих конструкций для позиционирования. Требуется по плану текущей расстановки конструкций, месту расположения платформы и величине  $k$  определить ячейки, до которых эта платформа сможет доставить комплектующие.

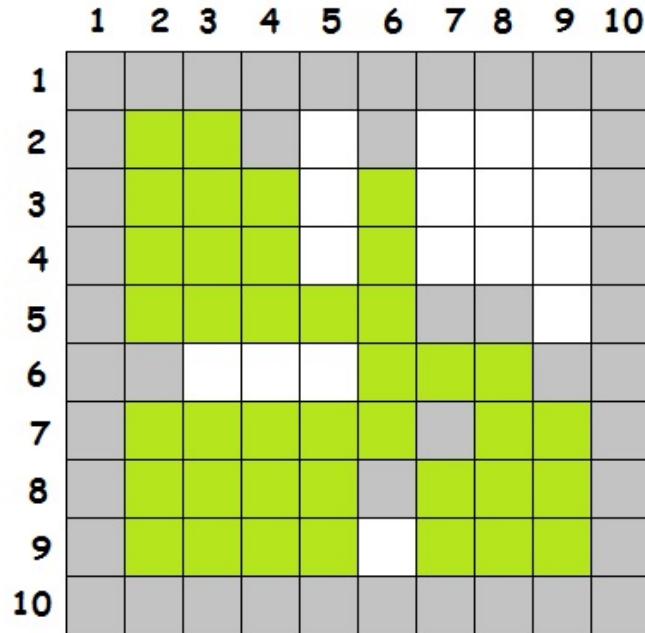
### ***Формат входных данных***

В первой строке содержатся три числа  $n$ ,  $m$  и  $k$  через пробел — размеры помещения и максимальное расстояние для позиционирования.  $1 \leq n, m, k \leq 100$ .

Далее в  $n$  строках по  $m$  символов в каждой представлен план помещения. Производственные ячейки (доступные для перемещения платформы) обозначены символом «.» (точка), ячейки, занятые светоотражающими конструкциями (недоступные для перемещения платформы) обозначены символом «#». Помещение по всему периметру огорожено этими конструкциями. Начальное расположение платформы обозначено символом «A».

### ***Формат выходных данных***

Вывести план того же помещения, где вместо точки вывести символ «+», если в эту ячейку можно доставить платформой из ее исходного положения комплектующие, и оставить точку, если нельзя. Начальное положение платформы должно быть обозначено символом «+». Символы «#» остаются без изменений.



### Примеры

#### Пример №1

##### Стандартный ввод

```
10 10 2
#####
#A.#.#...#
#. .... .#
#. .... .#
#. .... ##.#
##. .... ###
#. .... #..#
#. .... #..#
#. .... .#
#####
```

##### Стандартный вывод

```
#####
#++#. #...#
#+++. +...#
#+++. +...#
#+++++##. #
##...+++##
#+++++##++#
#+++++##++#
#+++++. +###
#####
```

### *Комментарий к примеру 1*

В приведенном тесте покажем, как платформа из  $(2, 2)$  сможет попасть в позицию  $(5, 2)$ , если  $k = 2$ : Из позиции  $(2, 2)$  сделать два шага вниз, ориентируясь на конструкцию  $(1, 2)$ , попадаем в позицию  $(4, 2)$ .

Из позиции  $(4, 2)$  сделать два шага вправо, ориентируясь на конструкцию  $(4, 1)$ , попадаем в позицию  $(4, 4)$ .

Из позиции  $(4, 4)$  сделать один шаг вниз, ориентируясь на конструкцию  $(2, 4)$ , попадаем в позицию  $(5, 4)$ .

Из позиции  $(5, 4)$  сделать два шага влево, ориентируясь на конструкцию  $(5, 1)$ , попадаем в позицию  $(5, 2)$ .

Обратите внимание, что из достижимой позиции  $(6, 6)$  нельзя переместиться в позицию  $(6, 5)$ , ориентируясь в начале шага на конструкцию  $(6, 9)$ , а в конце шага на конструкцию  $(6, 2)$ .

### *Решение*

Эта задача является достаточно стандартной задачей на 4-связном графе, заданном таблицей. Необходимо, выполняя условие на визуальный контакт платформы со светоотражающей конструкцией на заданном расстоянии, выяснить, до каких ячеек можно добраться из исходной. Построим функцию, которая вычислит расстояние от заданной ячейки до ближайшей конструкции в заданном направлении простым перебором ячеек на соответствующем пути. Далее для перемещения в соседнюю ячейку выясним по двум направлениям (вперед и назад относительно направления движения), каково это расстояние. Если оно соответствует заданному в условии (назад не более  $k - 1$  или вперед не более  $k$ , достаточно выполнения хотя бы одного из этих двух условий), то этот шаг можно сделать, отметив при этом новую ячейку как достижимую. Осталось произвести либо обход в глубину, либо обход в ширину и отметить все ячейки, достижимые из исходной, и далее вывести ответ в нужном формате.

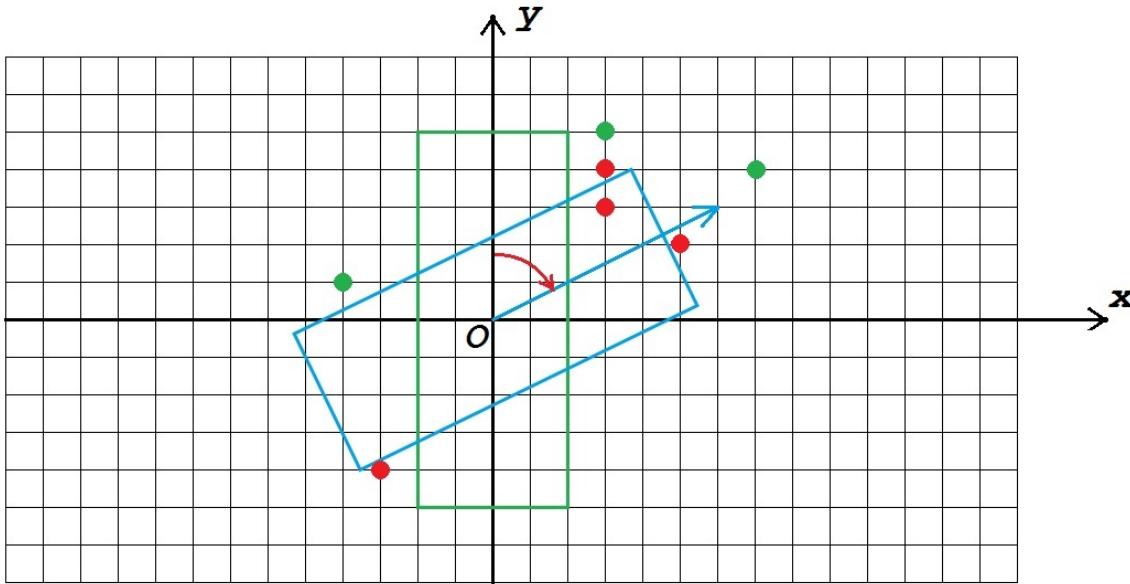
### *Задача III.1.1.4. Автономное гусеничное транспортное средство (30 баллов)*

Идея создать автономное средство передвижения рассматривается не только применительно к легковым или грузовым колесным автомобилям. Гусеничная техника тоже может быть очень полезна в автономном варианте. На данный момент идет тестирование опытного образца такой техники. Отрабатывается поворот на месте на заданный угол.

Испытательный полигон соотнесен с декартовой системой координат. Центр транспорта, который по форме представляет собой прямоугольник, расположен в начале координат. Изначально транспорт стоял так, что вектор его направления был ориентирован в положительном направлении оси  $oy$  (зеленый прямоугольник на рисунке). Далее он произвел поворот на месте так, что его центр остался в начале координат, а направляющий вектор стал равен  $(x, y)$  (синий прямоугольник на рисунке).

Перед началом поворота для испытания датчиков в некоторых точках испытательного полигона с целыми координатами были выставлены дорожные конусы. При

повороте некоторые конусы оказались сбиты. Требуется определить количество сбитых при повороте конусов. Если конус в какой-то момент поворота оказывается на границе прямоугольника, то он считается сбитым.



### *Формат входных данных*

В первой строке содержится четыре целых числа через пробел  $L, W, x, y$ . Первые два — это линейные размеры транспортного средства по оси  $ox$  и оси  $oy$  соответственно,  $2 \leq L, W \leq 100$ , оба эти параметра четные. Вторые два задают направляющий вектор после проведения поворота,  $-100 \leq x, y \leq 100$ ,  $x$  и  $y$  одновременно не равны 0. Из двух вариантов поворота выбирается тот, при котором поворот происходит на меньший угол.

Во второй строке содержится число  $n$  — количество установленных конусов.  $1 \leq n \leq 100$ .

Далее следует  $n$  строк, в каждой содержится два целых числа — координаты очередного конуса  $x_{cone_i}, y_{cone_i}$ . Эти координаты в пределах от  $-100$  до  $100$  включительно. Конусы находятся вне прямоугольника изначального положения транспорта. Для упрощения конусы следует считать точками.

### *Формат выходных данных*

Вывести одно число — количество сбитых при повороте конусов.

## Примеры

### Пример №1

Стандартный ввод
4 10 6 3
7
-4 1
-3 -4
3 3
3 4
3 5
5 2
7 4
Стандартный вывод
4

### Комментарий к примеру 1

На рисунке красными выделены позиции конусов, которые при повороте оказались сбиты. Зеленым обозначены конусы, которые не были сбиты.

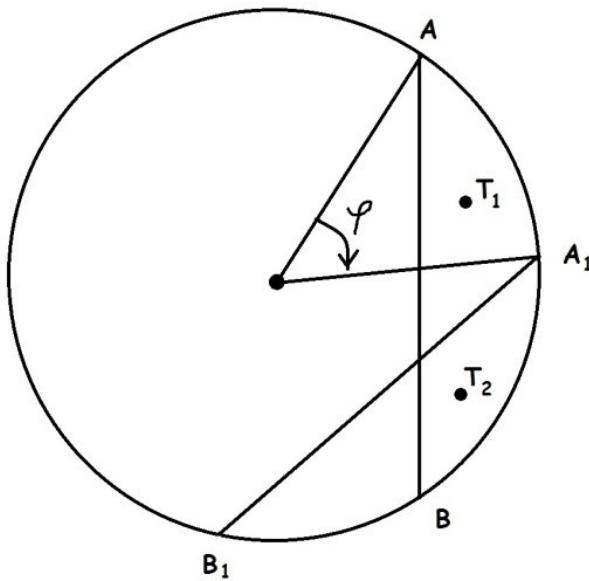
### Решение

Для решения задачи нужно уметь вращать точку на плоскости относительно начала координат на заданный угол. Напомним, что при повороте на угол  $\phi$  новые координаты  $(X_n, Y_n)$  вычисляются из старых  $(X, Y)$  по формулам:

$$X_n = X \cdot \cos \phi - Y \cdot \sin \phi;$$

$$Y_n = X \cdot \sin \phi + Y \cdot \cos \phi.$$

Помимо этого, нужно знать, как по двум точкам плоскости построить общее уравнение прямой  $A \cdot x + B \cdot y + C = 0$ , проходящей через них. Основное свойство этого уравнения в том, что при подстановке в него вместо  $x$  и  $y$  координат точки плоскости по знаку полученного результата можно выяснить, из какой полуплоскости эта точка. Более точно это позволяет узнать, находятся ли две разные точки в разных полуплоскостях или в одной относительно этой прямой. Далее транспорт будем рассматривать как соответствующий прямоугольник, а конусы как точки.



Для каждого конуса вычислим расстояние от него до центра прямоугольника. Если это расстояние больше половины диагонали прямоугольника, то эту точку транспорт не сбьет. Построим окружность с центром в начале координат и радиусом, равным половине диагонали. Допустим, расстояние до точки конуса меньше либо равно половине диагонали. Тогда, если этот конус (точка  $T_1$ ) будет сбит, то только той стороной прямоугольника, которая отсекает от окружности сегмент, в котором эта точка находится. Пусть до поворота эта сторона прямоугольника при пересечении с окружностью вращения обозначена  $AB$ , а после поворота  $A_1B_1$ . По точкам  $A$  и  $B$  при помощи формул вращения вычислим точки  $A_1$  и  $B_1$ . На приведенном рисунке мы обозначили только эту одну сторону прямоугольника, на самом деле их четыре. Точка  $T_1$  будет сбита, а точка  $T_2$ , которая находится в том же сегменте, что и точка  $T_1$ , не будет сбита при повороте на угол  $\phi$ . Как это узнать? Подставим координаты точки в общие уравнения прямой  $AB$  и прямой  $A_1B_1$ . Кроме того, подставим в эти уравнения и начало координат. Изначально, так как отрезок  $AB$  отсекал точку сегмента от начала координат, то знаки результатов подстановок начала координат и этой точки будут разные. Если точка оказалась сбита, то после поворота она будет в одной полуплоскости с началом координат относительно прямой  $A_1B_1$ , если не сбита, то так и останется в разных полуплоскостях. Проведя эти вычисления для всех точек и для соответствующих им отрезков—сторон прямоугольника, подсчитаем, сколько точек будет сбито. Не забудем, что если точка окажется на стороне  $A_1B_1$  или на окружности в области смены знака, то она считается сбитой.

### **Задача III.1.1.5. Размыкая циклы (30 баллов)**

Передовые технологии позволяют сделать то, что ранее считалось невозможным. Например, реализовать технологические цепочки, внутри которых есть циклические зависимости. Допустим, процесс  $A$  требует предварительного выполнения процесса  $B$ , а процесс  $B$  может быть выполнен только после выполнения процесса  $A$ , то есть получается замкнутый цикл. Но если задействовать новые аддитивные технологии, то оба процесса можно делать параллельно, разбив каждый на множество мелких слоев, избавившись, таким образом от циклической зависимости. Будем считать, что

время выполнения объединенного процесса, в результате которого выполнится и  $A$ , и  $B$  будет равно удвоенной сумме времен выполнения процессов  $A$  и  $B$  по отдельности, так как происходит регулярная смена выполняемых процессов. Аналогично, если есть несколько процессов, циклически зависящих друг от друга, данная технология позволяет выполнить их одновременно, но с коэффициентом, равным количеству этих процессов.

Дан набор из  $n$  технологических процессов  $a_i$ . Для каждого процесса известно время его выполнения  $t_i$ , а также список процессов, которые должны быть выполнены до того, как процесс  $a_i$  можно начать выполнять. Если есть циклические зависимости, то они могут быть выполнены в рамках интегрированных аддитивных технологий послойно. Общее время выполнения такого замкнутого подмножества операций равно сумме времен выполнения каждого ее элемента, умноженной на общее количество элементов в ней. При этом все процессы, от которых эта циклическая цепочка зависит, должны быть выполнены ранее.

Так как весь проект является экспериментальным, он требует пристального мониторинга, то есть никакие два процесса (за исключением циклически зависимых) не могут выполняться одновременно.

Требуется по этим данным выяснить, за какое минимальное время можно в этих условиях реализовать процесс с номером  $z$ .

### ***Формат входных данных***

В первой строке содержится два целых числа  $n$  и  $z$  через пробел — общее количество технологических процессов — и номер интересующего процесса.  $1 \leq n \leq 10^5$ ,  $1 \leq z \leq n$ .

Во второй строке содержится  $n$  целых чисел  $t_i$  через пробел — время выполнения процесса  $a_i$ .  $1 \leq t_i \leq 10^6$ .

Далее следуют  $n$  строк, описывающих зависимости. В  $i$ -й из этих строк содержится список из  $k_i$  процессов, которые должны быть выполнены до того, как начнется выполнения процесса  $a_i$ . Список начинается числом  $k_i$  и далее следуют  $k_i$  чисел через пробел — номера процессов, от которых зависит процесс  $a_i$ . Числа в строке не повторяются и не равны  $a_i$ . Все они в пределах от 1 до  $n$ .

Сумма всех чисел  $k_i$  не превосходит 200000.

### ***Формат выходных данных***

Вывести одно число — минимальное время, требуемое для реализации процесса с номером  $z$ .

## Примеры

### Пример №1

Стандартный ввод
8 5 1 2 3 3 2 2 1 4 2 2 8 1 3 2 2 6 2 3 7 2 1 2 0 2 5 6 2 5 7
Стандартный вывод
44

## Решение

В задаче нужно найти все компоненты сильной связности ориентированного графа, для каждой подсчитать ее размер и, исходя из этого, узнать суммарное время выполнения соответствующего набора процессов. Для этого воспользуемся серией обходов графа в глубину. Первым обходом в глубину  $dfs1$  для каждой вершины графа занесем ее в некоторый набор  $order$  при выходе из нее  $dfs1$ . Далее построим транспонированный граф, в котором все дуги ориентированы в обратную сторону. Переберем вершины этого транспонированного графа в порядке, обратном порядку  $order$ , и при нахождении очередной в этом порядке непокрашенной вершины, увеличиваем счетчик компонента сильной связности. Запускаем из этой вершины  $dfs2$ , который красит все достижимые из нее в транспонированном графе вершины в этот цвет. Все покрашенные в один цвет вершины исходного графа образуют компоненту сильной связности. Для каждой вершины исходного графа теперь можно узнать, в компоненте какой мощности она находится. Теперь запустим из целевой вершины  $z$  в исходном графе  $dfs3$ , который для каждой достижимой из  $z$  вершины добавляет ее стоимость (время исполнения соответствующего процесса), умноженную на величину ее компоненты сильной связности. В итоге получим ответ на задачу.

## Физика. 8–9 классы

### Задача III.1.2.1. Кинематика (20 баллов)

На круговой железной дороге длиной  $L = 6$  км расположена одна станция. От этой станции в направлении по часовой стрелке отъезжает автоматический локомотив, движущийся с постоянной скоростью  $v = 3$  м/с. Одновременно с этим между локомотивом и станцией начинает курсировать дрон, причем временем его остановок и разворотов можно пренебречь. Дрон движется туда и обратно строго над рельсами, при каждом развороте выбирая кратчайшую на момент разворота траекторию до следующего пункта назначения. Модуль скорости дрона всегда составляет  $u = 18$  м/с.

1. Какое наибольшее число раз в условиях задачи может дрон описать полную окружность, не меняя направления движения?
2. Какое расстояние пройдет дрон по часовой стрелке за время, за которое локомотив совершил полный оборот, если дополнительно известно, что через  $T = 10^3$  с после старта дрон двигался против часовой стрелки?
3. Тот же вопрос, если известно, что через  $T$  после старта дрон двигался по часовой стрелке.

#### Решение

Общее время движения поезда составляет

$$\tau = L/v.$$

За это время дрон проходит путь

$$S = u\tau = L \frac{u}{v}.$$

Очевидно, что при первых вылетах, когда локомотив находится недалеко от станции, дрон летит к нему по часовой стрелке и от него против часовой (см. рисунок III.1.1). Также легко понять, что когда локомотив завершает круг, дрон вылетает от станции против часовой стрелки и возвращается по часовой. Граница этих ситуаций проходит по точке, диаметрально противоположной станции, которую поезд как раз достигает в момент времени  $T$ .

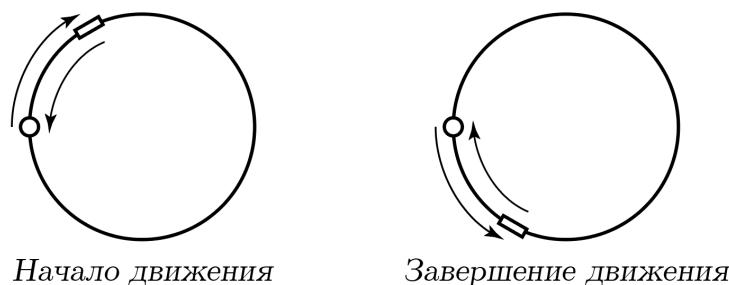


Рис. III.1.1

Когда поезд оказывается в этой точке, дрон движется по направлению от него (против часовой стрелки) — значит, он сделал свой последний вылет от станции по часовой, вернется против часовой, а следующий вылет уже произведет в обратном порядке. Таким образом, каждому отрезку пути, который дрон проделал в одном направлении, соответствовал обратный путь в другом направлении, поэтому дрон не сделает ни одного полного оборота без смены направлений, а его путь по часовой стрелке во втором пункте будет равен:

$$S_1 = \frac{S}{2} = \frac{Lu}{2v} = 18 \text{ км.}$$

Если же в данной точке дрон движется к поезду, то есть по часовой стрелке — значит он вылетел когда поезд еще не доехал до диаметральной к станции точки, но будет возвращаться уже за этой точкой. Поэтому в данном вылете дрон начнет и закончит движение в одном направлении: по часовой стрелке. При этом он сделает один полный оборот без перемены направления, а его общий путь в третьем пункте может быть найден по формуле:

$$S_2 = L + \frac{u\tau'}{2},$$

где  $\tau'$  — время дрона в движении за вычетом одного полного круга:

$$\tau' = \frac{L}{v} - \frac{L}{u}.$$

Таким образом, окончательно:

$$S_2 = \frac{L(u+v)}{2v} = 21 \text{ км.}$$

### Ответ:

1. 1.
2.  $S_1 = \frac{Lu}{2v} = 18 \text{ км.}$
3.  $S_2 = \frac{L(u+v)}{2v} = 21 \text{ км.}$

### *Критерии оценивания*

Замечено, что дрон может сделать или не сделать полный оборот.	1 балла
Замечено, что часть движения, пройденная по часовой стрелке, зависит от положения локомотива на середине пути.	2 балла
Найдено общее время дрона в движении.	2 балла
Дан правильный ответ на первый вопрос задачи.	4 балла
Дан правильный ответ на второй вопрос задачи.	5 баллов
Дан правильный ответ на третий вопрос задачи.	6 баллов
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

### *Задача III.1.2.2. Теплота (20 баллов)*

В ходе химического производства в калориметр налили  $m$  килограмм реагента при температуре  $t_0 = 30^\circ\text{C}$  и начали нагревать при помощи нагревателя постоянной мощности. Через время  $\tau = 10$  мин температура реагента поднялась на  $\Delta t = 50^\circ\text{C}$ , после чего в калориметр добавили  $2m$  второго реагента, имевшего начальную температуру  $t_2 = t_0 - \Delta t$ , в результате чего температура смеси быстро вернулась к отметке  $t_0$ . Через  $3\tau$  температура смеси достигла отметки, при которой вещества вступили в эндотермическую реакцию, в результате чего температура смеси вновь упала до  $t_0$  и образовался продукт, удельная теплоемкость которого равна среднему арифметическому удельных теплоемкостей реагентов.

- При какой температуре  $t_3$  вещества вступают в реакцию?
- Через какое время после прохождения реакции температура смеси снова достигнет  $t_1 = t_0 + \Delta t$ , если вещества прореагировали полностью?
- До какой минимальной температуры  $t_4$  должен был быть нагрет первый реагент перед добавлением второго, чтобы вещества полностью прореагировали сразу после смешивания?

### *Решение*

Обозначим удельную теплоемкость первого реагента  $c_1$ , второго —  $c_2$ , продукта —  $c_3 = (c_1 + c_2)/2$  (известно из условий). Мощность нагревателя обозначим  $P$ . На основании условий можно записать следующие уравнения теплового баланса:

$$P\tau = mc_1\Delta t;$$

$$mc_1\Delta t = 2mc_2\Delta t;$$

$$3P\tau = (mc_1 + 2mc_2)(t_3 - t_0).$$

Из второго уравнения системы легко получить:

$$2mc_2 = mc_1.$$

Подставляя этот результат в третье уравнение и сравнивая его с первым, получим:

$$3me_1\Delta t = 2me_1(t_3 - t_0).$$

Откуда можно получить:

$$t_3 = t_0 + \frac{3}{2}\Delta t = 105^\circ\text{C}.$$

Для ответа на второй пункт задачи достаточно записать еще одно уравнение теплового баланса (приняв во внимание закон сохранения массы и полученные ранее соотношения между теплоемкостями):

$$P\tau' = 3mc_3\Delta t = \frac{3}{2}m(c_1 + c_2)\Delta t = \frac{3}{2}mc_1 \left(1 + \frac{1}{2}\right) \Delta t.$$

Из первого уравнения теплового баланса выразим мощность:

$$P = \frac{mc_1\Delta t}{\tau}$$

и подставим в последнее:

$$me_1\Delta t \frac{\tau'}{\tau} = \frac{9}{4}me_1\Delta t,$$

откуда непосредственно:

$$\tau' = \frac{9}{4}\tau = 22,5 \text{ мин.}$$

Для выполнения требований последнего пункта необходимо, чтобы после смешивания реагентов их температура упала до  $t_3$ . Запишем соответствующее уравнение теплового баланса:

$$mc_1(t_4 - t_3) = 2mc_2(t_3 - t_2)$$

и учтем соотношение  $2mc_2 = mc_1$ :

$$me_1(t_4 - t_3) = me_1(t_3 - t_2).$$

Подставим известные значения  $t_3, t_2$ :

$$t_4 = 2t_3 - t_2 = 2t_0 + 3\Delta t - (t_0 - \Delta t) = t_0 + 4\Delta t = 230^\circ\text{C}.$$

**Ответ:**

1.  $t_3 = t_0 + \frac{3}{2}\Delta t = 105^\circ\text{C}.$
2.  $\tau' = \frac{9}{4}\tau = 22,5 \text{ мин.}$
3.  $t_4 = t_0 + 4\Delta t = 230^\circ\text{C}.$

### Критерии оценивания

Верно записано хотя бы одно уравнение теплового баланса.	2 балла
Верно записаны все необходимые для решения первого пункта уравнения теплового баланса.	еще 3 балла
Верно записана связь теплоты с мощностью.	2 балла
Дан правильный ответ на первый вопрос задачи.	3 балла
Дан правильный ответ на второй вопрос задачи.	5 баллов
Дан правильный ответ на третий вопрос задачи.	5 баллов
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

### Задача III.1.2.3. Электрические цепи (20 баллов)

На изолирующую подложку фотолитографически нанесен цветок из тонких дорожек проводника, форма которого представлена на рисунке III.1.2. Все отрезки проводника идентичны и имеют сопротивление  $r = 2,5 \Omega$  каждый.

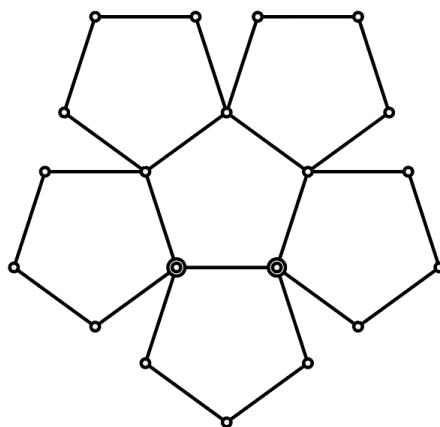


Рис. III.1.2

- Определите сопротивление  $R$  между контактами, выделенными на рисунке дополнительной обводкой.
- При подключения источника к этим контактам в цветке возник ток с максимальной величиной  $I_{max} = 32 \text{ mA}$  на одном отрезке. Чему равно минимальное значение тока  $I_{min}$  на одном отрезке цепи?
- Сколько существует способов соединить пару любых контактов (отмечены на рисунке маленькими кружками) дополнительным проводом с малым сопротивлением так, чтобы сопротивление между выделенными точками не изменилось?

### Решение

Прежде всего, заметим, что изображенная схема является самоподобной: она может быть представлена в виде пяти резисторов  $R'$ , подключенных «пятиугольником» (см. рисунок III.1.3), каждый из которых в свою очередь представляет пять резисторов  $r$ , подключенных таким же пятиугольником.

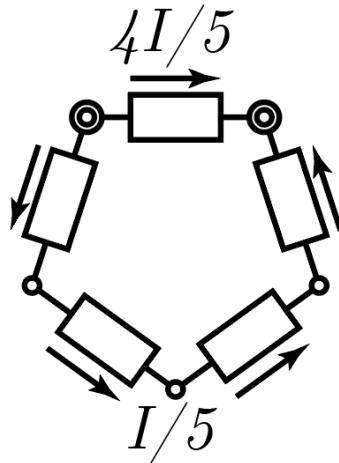


Рис. III.1.3

Для этой схемы легко найти как эквивалентное сопротивление (она представляет собой один резистор, подключенный параллельно набору из четырех последовательно соединенных таких же):

$$\frac{1}{R'} = \frac{1}{r} + \frac{1}{4r} = \frac{5}{4r} \Rightarrow R' = \frac{4}{5}r,$$

так и распределение токов (изображено на рисунке III.1.3) из того соображения, что на параллельных элементах цепи одинаковые напряжения:

$$U = I_{\text{верх}}r = I_{\text{низ}} \cdot 4r = IR',$$

где  $I_{\text{верх}}$  — ток через верхнюю ветвь цепи (один резистор),  $I_{\text{низ}}$  — ток через нижнюю часть цепи (четыре резистора),  $I$  — полный ток между контактами.

Таким образом, для полного сопротивления цепи легко записать:

$$R = \frac{4}{5}R' = \frac{4}{5} \cdot \frac{4}{5}r = \frac{16}{25}r = 1,6 \text{ Ом.}$$

Как можно видеть из рисунка, минимальный и максимальный ток в пятиугольной цепи относятся как 1 : 4. Поскольку схема самоподобна, это отношение повторяется дважды. Таким образом, во всем цветке максимальный ток будет достигаться на участке, непосредственно соединяющем контакты, а минимальный — на каждом из четырех последовательно соединенных отрезков каждого из четырех последовательно соединенных лепестков, причем:

$$I_{\min} = \frac{I_{\max}}{16} = 2 \text{ мА.}$$

Для ответа на последний вопрос найдем падение напряжения между левым выделенным контактом и всеми остальными контактами (сделать это очень просто после

того, как мы установили распределение токов). Для краткости записи обозначим  $U$  минимальное падение напряжения  $U = I_{min}r$ :

Из рисунка III.1.4 видно, что в схеме только две пары контактов, напряжение между которыми равно нулю (соответствующие значениям  $4U$ ,  $8U$  и  $12U$ ). Если соединить любую из этих пар — ток по дополнительному проводу не пойдет и общее сопротивление цепи не изменится.

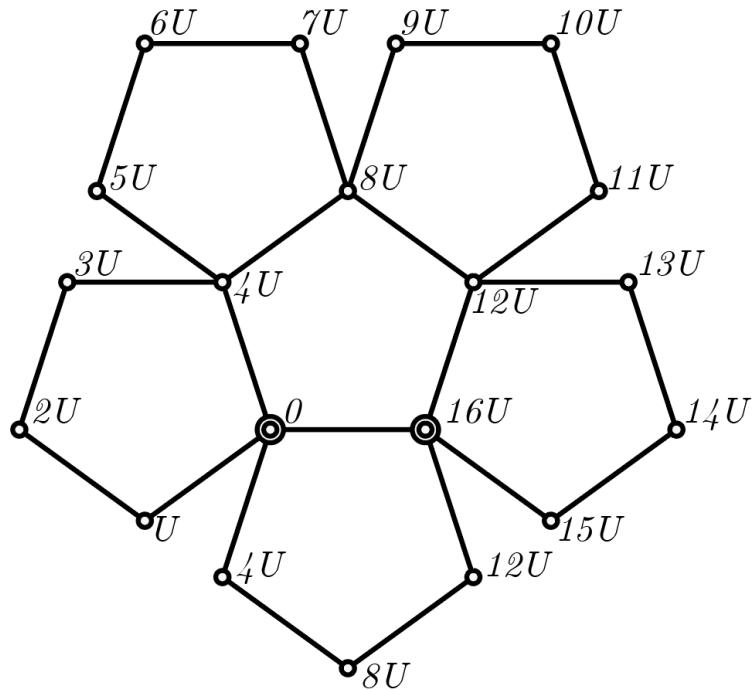


Рис. III.1.4

**Ответ:**

1.  $R = \frac{16}{25}r = 1,6 \text{ Ом.}$
2.  $I_{min} = \frac{I_{max}}{16} = 2 \text{ мА.}$
3. 2.

### *Критерии оценивания*

Найдено эквивалентное сопротивление одного пятиугольника или рассуждение, выполняющее ту же роль в решении.	3 балла
Показано, что схема может быть представлена как пятиугольник пятиугольников.	3 балла
Найдено распределение токов в одном пятиугольнике.	2 балла
Указано, что чтобы сопротивление цепи не изменилось, напряжение между соединяемыми контактами должно быть равно нулю.	2 балла
Дан правильный ответ на первый вопрос задачи.	3 балла
Дан правильный ответ на второй вопрос задачи.	3 балла
Дан правильный ответ на третий вопрос задачи.	4 балла
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

### *Задача III.1.2.4. Оптика (20 баллов)*

Для очистки поверхности металла используется промышленный лазер, выдающий пучок параллельных лучей радиусом  $R = 6$  мм. В излучении лазера присутствуют видимая и инфракрасная составляющие, причем мощность видимого излучения составляет  $P_v = 4$  Вт, а инфракрасного  $P_{ir} = 2$  Вт. Излучение фокусируется при помощи тонкой линзы, плоскость которой перпендикулярна световому пучку. Из-за дисперсии фокусное расстояние линзы для видимого излучения  $F_v = 10$  см, а для инфракрасного  $F_{ir} = 15$  см.

1. Определите радиус сфокусированного пучка в его самом узком месте, считая пучком все точки пространства, до которых доходит хотя бы одна компонента излучения.
2. Считая, что мощность каждой компоненты равномерно распределена по всему сечению пучка, определите, во сколько раз будут отличаться мощности двух компонент излучения, попадающие на точечный образец, расположенный ровно в середине между двумя фокусами системы.
3. При калибровке лазера лаборант установил перпендикулярно главной оптической оси линзы круглый образец, совместив его центр с точкой, в которой лазерный пучок выглядит максимально сфокусированным. При каком минимальном радиусе образца на него попадет все излучение лазера?

### Решение

Прежде всего, изобразим ход лучей видимого (сплошные линии) и инфракрасного (пунктирные линии) диапазонов в линзе (рисунок III.1.5).

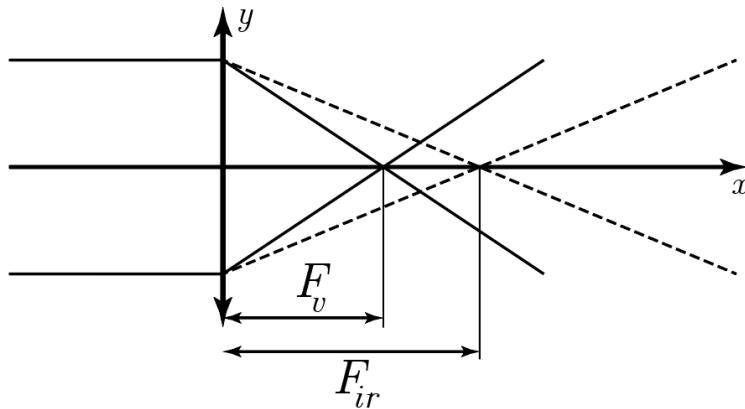


Рис. III.1.5

Наиболее узкое место пучка достигается в точке, в которой пересекаются расходящиеся лучи видимой и сходящиеся лучи инфракрасной компоненты. Координаты этой точки могут быть найдены любым корректным геометрическим или алгебраическим способом. Мы используем алгебраический подход, введя оси координат  $0x$  вдоль ГОО линзы и  $0y$  в плоскости линзы с началом в ее оптическом центре (см. рисунок III.1.5).

В такой системе координат крайние преломленные видимые лучи описываются уравнениями прямых:

$$y = \pm R \mp \frac{R}{F_v} x,$$

а аналогичные инфракрасные —

$$y = \pm R \mp \frac{R}{F_{ir}} x.$$

Используя уравнение идущего вверх луча одной компоненты и идущего вниз луча другой, получим систему, решением которой для координаты  $y$  является искомый радиус пучка в самом узком месте:

$$\begin{cases} y = -R + \frac{R}{F_v} x \\ y = R - \frac{R}{F_{ir}} x \end{cases}$$

Исключая из системы уравнений  $x$ , окончательно получим

$$y = R \left( \frac{F_{ir} - F_v}{F_{ir} + F_v} \right) = 1,2 \text{ мм.}$$

Для ответа на второй вопрос заметим, что при равномерном распределении мощности по сечению пучка интенсивность (плотность потока энергии) может быть найдена как

$$I = P/S,$$

где  $S = \pi R'^2$  — площадь пучка в области, где он имеет радиус  $R'$ .

Поскольку образец в условиях этого пункта является точечным, попадающая на него полная мощность каждой компоненты зависит только от ее интенсивности:

$$\frac{P'_v}{P'_{ir}} = \frac{I_v}{I_{ir}} = \frac{P_v}{P_{ir}} \cdot \left( \frac{R_{ir}}{R_v} \right)^2.$$

Используя уравнения для лучей, полученные выше, определим, что в точке посередине между двумя фокусами,  $x$  координата которой равна

$$x = \frac{F_v + F_{ir}}{2},$$

радиус  $R_v$  видимого пучка составляет

$$R_v = R \left( 1 - \frac{F_v + F_{ir}}{2F_v} \right) = \frac{R(F_v - F_{ir})}{2F_v},$$

а инфракрасного

$$R_{ir} = \frac{R(F_v - F_{ir})}{2F_{ir}}.$$

Однаковые числители этих дробей сокращаются при поиске их отношения:

$$\frac{R_{ir}}{R_v} = \frac{F_v}{F_{ir}}.$$

Окончательно

$$\frac{P'_v}{P'_{ir}} = \frac{P_v}{P_{ir}} \cdot \left( \frac{F_v}{F_{ir}} \right)^2 \approx 0,89.$$

Для ответа на последний вопрос задачи заметим, что пучок выглядит максимально сфокусированным в точке  $F_v$ , поскольку только видимая компонента излучения может быть воспринята глазом. Образец, помещенный в эту точку, получает всю мощность видимой части пучка, но только часть инфракрасной, пропорциональную доле площади этого пучка, попадающей на образец. В данной точке радиус  $r_{ir}$  инфракрасной части находится из тех же уравнений прямых:

$$r = R \left( 1 - \frac{F_v}{F_{ir}} \right) = 2 \text{ мм.}$$

Этот радиус и есть минимальный радиус образца, при котором на него попадут целиком обе компоненты излучения.

**Ответ:**

1.  $y = R \left( \frac{F_{ir} - F_v}{F_{ir} + F_v} \right) = 1,2 \text{ мм.}$
2.  $\frac{P'_v}{P'_{ir}} = \frac{P_v}{P_{ir}} \cdot \left( \frac{F_v}{F_{ir}} \right)^2 \approx 0,89$  или  $\frac{P'_{ir}}{P'_v} = \frac{P_{ir}}{P_v} \cdot \left( \frac{F_{ir}}{F_v} \right)^2 \approx 1,13.$
3.  $r = R \left( 1 - \frac{F_v}{F_{ir}} \right) = 2 \text{ мм.}$

### *Критерии оценивания*

Верно изображен ход лучей в линзе.	2 балла
Составлены уравнения прямых, соответствующие ходу лучей или эквивалентные геометрические построения.	3 балла
Указано, что мощность, получаемая точечным источником прямо пропорциональна плотности мощности (интенсивности) соответствующей компоненты излучения.	3 балла
Понято, что точка, в которой пучок выглядит наиболее сфокусированным, является фокусом видимой компоненты излучения.	2 балла
Дан правильный ответ на первый вопрос задачи.	3 балла
Дан правильный ответ на второй вопрос задачи.	4 балла
Дан правильный ответ на третий вопрос задачи.	3 балла
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

### *Задача III.1.2.5. Динамика (20 баллов)*

Полная стартовая масса ракеты «Протон-М» составляет  $M = 705$  т, из которых  $M_I = 460$  т приходится на первую ступень, причем  $m_I = 430$  т из этой массы занимает топливо. Все топливо первой ступени равномерно сгорает за  $\tau = 2$  мин и выбрасывается с постоянной скоростью  $u = 2,8$  км/с относительно ракеты. Считать ускорение свободного падения  $g \approx 9,8$  Н/м.

1. Какова сила тяги двигателей первой ступени?
2. Какой вес имел космонавт перед стартом если сразу после старта ракеты его вес составлял  $P_0 = 2$  кН?
3. Какой вес имел тот же космонавт между отделением первой ступени и включением двигателей второй?

Влиянием сопротивления воздуха во всех пунктах задачи пренебречь.

### *Решение*

Рассмотрим два близких момента времени  $0$  и  $\Delta t$  при движении ракеты. Обозначим  $M_0$  массу ракеты в момент времени  $0$ ,  $v_0$  — ее скорость в этот же момент,  $\Delta m$  — массу порции топлива, выброшенной за время  $\Delta t$ ,  $\Delta v$  — приращение ее скорости за время  $\Delta t$ . Направив ось  $Oy$  вверх, можем записать в ИСО (относительно Земли) закон сохранения импульса:

$$(M_0 + \Delta m)v_0 = M_0(v_0 + \Delta v) + \Delta m(v_0 - u).$$

Сокращая  $M_0v_0$  и  $\Delta mv_0$  в обеих частях уравнения, получим:

$$0 = M_0\Delta v - \Delta mu.$$

Таким образом, приращение импульса ракеты  $M_0\Delta v$  равно произведению скорости реактивной струи относительно ракеты на массу израсходованного топлива  $\Delta mu$ . Используя второй закон Ньютона в импульсной форме, получим выражение для силы тяги двигателей первой ступени:

$$F = \frac{\Delta p}{\Delta t} = u \frac{\Delta m}{\Delta t} \approx 10 \text{ МН.}$$

Поскольку по условиям задачи топливо вытекает равномерно,

$$\frac{\Delta m}{\Delta t} = \frac{m_I}{\tau}.$$

Ускорение, создаваемое этой силой в каждый момент времени  $t$ , зависит от текущей массы ракеты  $M(t)$ :

$$a(t) = \frac{F - M(t)g}{M(t)} = \frac{F}{M(t)} - g.$$

В первые моменты после старта масса в этом выражении равна  $M$  и вес космонавта определяется соотношением:

$$P_0 = m_{\kappa}(g + a) = F \frac{m_{\kappa}}{M},$$

где  $m_{\kappa} = P_0M/F$  — масса космонавта.

Перед стартом космонавт находится в ИСО, и его вес равен

$$P = m_{\kappa}g = P_0 \frac{Mg}{F} = P_0 \frac{Mg}{um_I/\tau} \approx 1,4 \text{ кН.}$$

Между отделением первой ступени и включением двигателей второй ракеты находится в состоянии свободного падения, поэтому вес космонавта равен нулю.

**Ответ:**

1.  $F = u \frac{\Delta m}{\Delta t} \approx 10 \text{ МН.}$
2.  $P = P_0 \frac{Mg}{um_I/\tau} \approx 1,4 \text{ кН.}$
3.  $P_2 = 0.$

### Критерии оценивания

Записан закон сохранения импульса или уравнение Мещерского, позволяющие найти силу тяги двигателей ракеты.	4 балла
Показана связь веса космонавта с ускорением ракеты.	2 балла
Верно найдено ускорение ракеты.	3 балла
Дан правильный ответ на первый вопрос задачи.	3 балла
Дан правильный ответ на второй вопрос задачи.	5 балла
Дан правильный ответ на третий вопрос задачи.	3 балла
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

## Физика. 10–11 классы

### Задача III.1.3.1. Статика (20 баллов)

Для регулировки высоты станка используется устройство, изображенное на рисунке III.1.6, состоящее из оси ( $O$ ), на которую одета подвижная втулка ( $B$ ), соединенная с поршнем ( $\Pi$ ). Расстояние от нижнего шарнира до оси  $l = 60$  см, коэффициент трения между осью и втулкой  $\mu = 0,1$ . Считать ускорение свободного падения  $g \approx 9,8 \text{ м/с}^2$ .

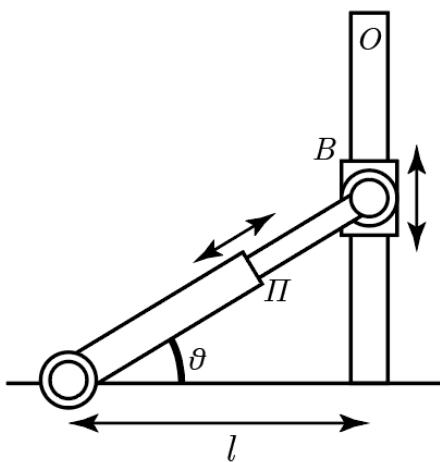


Рис. III.1.6

- При каком минимальном угле  $\theta$  может функционировать ненагруженный подъемник, если трение в шарнирах пренебрежимо мало?
- Подъемник может удерживать от самопроизвольного опускания закрепленный

на втулке груз максимальной массы  $M = 400$  кг при угле  $\theta_0 = 15^\circ$ . Какой максимальный груз он может поднимать из этого положения?

3. В результате утечки масла в нижнем подшипнике возникло трение с коэффициентом  $2\mu$ . Как изменится ответ на вопрос первого пункта если радиус подшипника равен  $r = 5$  см?

*Указание:* в последнем пункте считать справедливым приближение малых углов  $\sin \theta \approx \theta$ ;  $\cos \theta \approx 1 - \theta^2/2$ .

### Решение

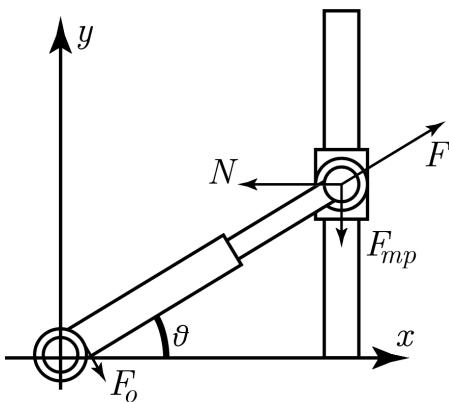


Рис. III.1.7

Введем систему координат как изображено на рисунке III.1.7 (силы на рисунке соответствуют третьему пункту задачи). Давление в поршне создает действующую на втулку силу  $F$ , направленную вдоль поршня. Вдоль оси  $0x$  ей противодействует сила нормальной реакции оси, а вдоль оси  $0y$  — сила трения. При этом из первого условия равновесия втулки вдоль  $0x$  всегда можно записать:

$$F \cos \theta - N = 0,$$

а для минимального угла, при котором поршень еще может преодолеть трение в оси можно записать в проекции на  $0y$ :

$$F \sin \theta - \mu N = 0.$$

Эта система уравнений тождественна системе, описывающей равновесие тела на наклонной плоскости. Решая ее, получим:

$$\theta = \operatorname{arctg} \mu \approx 5,7^\circ.$$

При решении второго пункта выражение для силы нормальной реакции остается прежним:

$$N = F \cos \theta,$$

но в проекции на  $0y$  появляется дополнительная сила тяжести. При этом при удержании максимального груза  $M$  сила трения  $\mu N = \mu F \cos \theta$  будет направлена вверх:

$$F \sin \theta_0 + \mu F \cos \theta_0 - Mg = 0,$$

а при подъеме максимального груза  $m$  — вниз:

$$F \sin \theta_0 - \mu F \cos \theta_0 - mg = 0,$$

Исключая из полученной системы уравнений  $F$ , получим

$$m = \frac{\sin \theta_0 - \mu \cos \theta_0}{\sin \theta_0 + \mu \cos \theta_0} M \approx 183 \text{ кг.}$$

При решении последнего пункта задачи условие равновесия поршня удобнее рассмотреть в виде уравнения моментов сил, действующих на поршень, записанного относительно оси его нижнего шарнира. В этом случае момент силы трения втулки об ось:

$$M_{\text{тр}} = \mu Nl,$$

момент силы нормальной реакции:

$$M_N = Nl \tan \theta.$$

Записанное с использованием этих моментов сил второе условие равновесия поршня (уравнение моментов) имеет вид:

$$Nl \tan \theta - \mu Nl = 0$$

и имеет такое же решение как более стандартный подход, приведенный в первом пункте. Однако, при возникновении дополнительного трения в оси  $F_o$ , необходимо также учесть момент этой силы трения. Данная сила будет равна:

$$F_o = 2\mu F = \frac{2\mu N}{\cos \theta}$$

и приложена по касательной к внешней поверхности подшипника, поэтому относительно оси вращения ее момент составит:

$$M_o = \frac{2\mu Nr}{\cos \theta}.$$

С учетом этой добавки уравнение моментов принимает вид

$$Nl \tan \theta - \mu Nl - \frac{2\mu Nr}{\cos \theta} = 0.$$

Домножим его на  $(\cos \theta)/N$ :

$$l \sin \theta - \mu l \cos \theta - 2\mu r = 0.$$

Используя приближение малых углов, это уравнение можно свести к квадратному:

$$l\theta - \mu l \left(1 - \frac{\theta^2}{2}\right) - 2\mu r = 0.$$

Единственный положительный корень этого уравнения:

$$\theta_1 = -\frac{1}{\mu} + \sqrt{\frac{1}{\mu^2} + \frac{2(l+2r)}{l}} \approx 0,115 \text{ рад} \approx 6,6^\circ.$$

Маленькое значение полученного угла свидетельствует о правомерности применения приближения малых углов.

**Ответ:**

1.  $\theta = \arctg \mu \approx 5,7^\circ$ .
2.  $m = \frac{\sin \theta_0 - \mu \cos \theta_0}{\sin \theta_0 + \mu \cos \theta_0} M \approx 183 \text{ кг.}$
3.  $\theta_1 = -\frac{1}{\mu} + \sqrt{\frac{1}{\mu^2} + \frac{2(l+2r)}{l}} \approx 0,115 \text{ рад} \approx 6,6^\circ$ .

### *Критерии оценивания*

Показано, что ограничения на угол функционирования ненагруженного подъемника обусловлены силой трения.	2 балла
Верно записан второй закон Ньютона или уравнение моментов без нагрузки.	3 балла
Верно записан второй закон Ньютона или уравнение моментов под нагрузкой.	2 балла
Верно указана точка приложения и направление силы трения в шарнире.	2 балла
Верно записано уравнение моментов с учетом трения в шарнире.	2 балла
Дан правильный ответ на первый вопрос задачи.	2 балла
Дан правильный ответ на второй вопрос задачи.	3 балла
Дан правильный ответ на третий вопрос задачи.	4 балла
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

### *Задача III.1.3.2. Термодинамика (20 баллов)*

Газотурбинный двигатель летательного аппарата, предназначенного для космических миссий, работает по циклу, состоящему из двух изобар и двух адиабат (см. рисунок III.1.8), используя в качестве рабочего тела забортную атмосферу. Известно, что работа, совершаемая на фазе выпуска 4 – 1, и теплота, выделяющаяся при сгорании топлива на фазе подвода тепла 2 – 3 определяются конструкцией двигателя и не зависят от внешних условий. На испытательном стенде в атмосфере Земли двигатель показал КПД  $\eta_0 = 35\%$ .

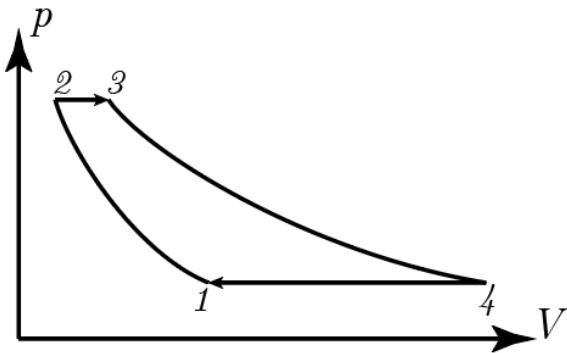


Рис. III.1.8

1. Каким будет КПД этого же двигателя в атмосфере планеты, состоящей целиком из одноатомных газов?
2. Определите эффективное число степеней свободы молекул атмосферы планеты, в которой КПД данного двигателя составил  $\eta_1 = 26\%$ .
3. Во сколько раз отличаются минимальное и максимальное давление в цикле в условиях земной атмосферы, если известно, что любой адиабатический процесс в ней описывается уравнением  $pV^{7/5} = \text{const}$ ?

### Решение

Поскольку процессы 1 – 2 и 3 – 4 адиабатические, КПД цикла полностью определяется тепловыми эффектами изобарных участков. При этом тепло подводится рабочему телу в процессе 2 – 3 и отводится в процессе 4 – 1:

$$\eta = 1 - \frac{Q_{41}}{Q_{23}}.$$

В согласии с первым началом термодинамики, теплота  $Q_{41}$  может быть выражена через работу на этом участке  $A_{41} = P_1(V_4 - V_1)$  и изменение внутренней энергии на нем же  $\Delta U_{41} = \frac{i}{2}P_1(V_4 - V_1)$ , где  $i$  – число степеней свободы рабочего газа:

$$Q_{41} = P_1(V_4 - V_1) + \frac{i}{2}P_1(V_4 - V_1) = \frac{i+2}{2}A_{41}.$$

Таким образом,

$$\eta = 1 - \left( \frac{i+2}{2} \right) \frac{Q_{23}}{A_{41}} \Rightarrow 1 - \eta = \left( \frac{i+2}{2} \right) \frac{Q_{23}}{A_{41}}.$$

По условиям задачи величина  $Q_{23}/A_{41}$  неизменна, поэтому различие КПД в разных условиях обусловлено разным числом степеней свободы рабочего тела. В атмосфере Земли преобладают двухатомные газы, поэтому число степеней свободы молекул воздуха можно считать равным 5. В одноатомной атмосфере оно будет равно 3. Таким образом,

$$1 - \eta = \frac{5}{7}(1 - \eta_0) \Rightarrow \eta = \frac{2}{7} + \frac{5}{7}\eta_0 \approx 54\%.$$

Отвечая на второй пункт задачи, выразим  $Q_{23}/A_{41}$  в явной форме:

$$\frac{Q_{23}}{A_{41}} = \frac{2}{7}(1 - \eta_0).$$

Тогда

$$\eta_1 = 1 - \frac{i+2}{7}(1 - \eta_0).$$

Откуда можно выразить эффективное число степеней свободы:

$$i = 7 \frac{1 - \eta_1}{1 - \eta_0} - 2 \approx 6.$$

Для ответа на последний вопрос выразим  $Q_{23}$ ,  $Q_{41}$  через давления и объемы:

$$\eta_0 = 1 - \frac{Q_{41}}{Q_{23}} = 1 - \frac{A_{41}}{A_{23}} = 1 - \frac{P_1 \Delta V_1}{P_2 \Delta V_2}.$$

Из уравнения адиабаты  $PV^{7/5} = const$  автоматически следует:

$$\frac{\Delta V_1}{\Delta V_2} = \left( \frac{P_2}{P_1} \right)^{5/7}.$$

Подставляя данное равенство в соотношение для КПД, получим:

$$\eta_0 = 1 - \left( \frac{P_2}{P_1} \right)^{2/7},$$

откуда можно выразить отношение давлений:

$$\frac{P_2}{P_1} = (1 - \eta_0)^{-7/2} \approx 4,5.$$

**Ответ:**

1.  $\eta = \frac{2}{7} + \frac{5}{7}\eta_0 \approx 54\%$ .
2.  $i = 7 \frac{1 - \eta_1}{1 - \eta_0} - 2 \approx 6$ .
3.  $P_2/P_1 = (1 - \eta_0)^{-7/2} \approx 4,5$ .

### *Критерии оценивания*

Верно записано определение КПД термодинамического цикла.	3 балла
Найдена связь между работой газа и полученной теплотой на изобарных участках.	3 балла
Явно указана или верно учтена двухатомность атмосферы Земли.	2 балла
Дан правильный ответ на первый вопрос задачи.	3 балла
Дан правильный ответ на второй вопрос задачи.	4 балла
Дан правильный ответ на третий вопрос задачи.	5 баллов
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

### **Задача III.1.3.3. Электростатика (20 баллов)**

В ультразвуковой ловушке, позволяющей парить, практически не испытывая воздействия сил в горизонтальной плоскости, подвешена электрически нейтральная золотая частица в виде шара радиусом  $R = 13 \text{ нм}$  и массой  $M = 1,66 \cdot 10^{-19} \text{ кг}$ . В направлении на центр частицы с одной стороны последовательно выпускают десять электронов с начальными скоростями  $v_0 = 400 \text{ км/с}$  каждый. Если электрон достигает частицы — он присоединяется к ней. Масса электрона  $m_e = 9,1 \cdot 10^{-31} \text{ кг}$ . Коэффициент пропорциональности в законе Кулона  $k = \frac{1}{4\pi\epsilon_0} = 9,0 \cdot 10^9 \text{ Н}\cdot\text{м}^2/\text{Кл}^2$ , заряд электрона  $e = 1,6 \cdot 10^{-19} \text{ Кл}$ .

- Сколько избыточных электронов останется на частице?
- Какую скорость приобретет изначально неподвижная частица в результате всех взаимодействий?
- Чему будет равен модуль максимальной силы электростатического отталкивания между частицей и электроном в этом процессе?

Считайте, что плотность «электронного газа» в металлической частице распределяется сферически симметрично после присоединения каждого добавочного электрона и практически не изменяется за время полета внешнего электрона.

### *Решение*

Первый электрон, выпущенный в сторону частицы, не встречает противодействия и в любом случае присоединяется к ней. Допустим,  $n$  электронов уже находятся на частице и  $n + 1$ -ый летит в ее сторону издалека. Чтобы достичь частицы, электрон должен преодолеть кулоновское отталкивание, то есть его кинетическая энергия

должна превышать потенциальную энергию кулоновского отталкивания в момент присоединения к частице. Запишем соответствующий закон сохранения энергии:

$$\frac{m_e v_0^2}{2} = k \frac{n e^2}{R}.$$

В данном случае мы считаем наночастицу неподвижной, поскольку ее масса пре- восходит массу электрона на 12 порядков. Следовательно, в момент присоединения электрона его кинетическая энергия может быть сколь угодно близка к нулю.

Решая данное уравнение относительно  $n$  и округляя результат вниз, получим максимальное число электронов, при которых еще один может достичь ее, а округляя вверх — число электронов, которое останется на частице в итоге:

$$n = \left\lceil \frac{m_e v_0^2 R}{2 k e^2} \right\rceil = 5.$$

Для ответа на второй пункт задачи, отметим, что каждое взаимодействие, в результате которого электрон присоединяется к частице, является абсолютно неупругим, а каждое взаимодействие, в результате которого он отталкивается — абсолютно упругим столкновением. Поскольку  $M \gg m_e$ , мы можем считать, что при каждом упругом столкновении электрон меняет свой импульс на  $2m_e v_0$ , а при каждом неупругом — на  $m_e v_0$ . Таким образом, полное изменение импульса электронов:

$$\Delta p = 5m_e v_0 + 5 \cdot 2m_e v_0 = 15m_e v_0.$$

Разделив эту величину на массу частицы, получим:

$$v = \frac{\Delta p}{M} = 15 \frac{m_e}{M} v_0 \approx 33 \text{ мкм/с.}$$

Для ответа на последний вопрос заметим, что все электроны после пятого фактически подлетают к частице на одинаковое расстояние (поскольку ее заряд дальше не изменяется, а скорость изменяется пренебрежимо слабо в сравнении со скоростью электрона), которое может быть найдено из закона сохранения энергии:

$$r = \frac{10 k e^2}{m_e v_0^2},$$

испытывая при этом действие силы Кулона:

$$F_1 = \frac{5 k e^2}{r^2} = \frac{m_e^2 v_0^4}{20 k e^2} \approx 4.6 \cdot 10^{-12} \text{ Н.}$$

В то же время, максимальная сила отталкивания до этого достигается непосредственно перед присоединением последнего (пятого) электрона и равна:

$$F_2 = \frac{4 k e^2}{R^2} \approx 5,5 \cdot 10^{-12} \text{ Н.}$$

Можно видеть, что вторая из этих сил больше. Следовательно, максимальная величина силы электростатического отталкивания достигается непосредственно перед столкновением пятого электрона с частицей и равна  $F_2$ .

**Ответ:**

1.  $n = \left\lceil \frac{m_e v_0^2 R}{2ke^2} \right\rceil = 5.$
2.  $v = 15 \frac{m_e}{M} v_0 \approx 33 \text{ мкм/с.}$
3.  $F_2 = \frac{4ke^2}{R^2} \approx 5,5 \cdot 10^{-12} \text{ Н.}$

### *Критерии оценивания*

Верно записан закон сохранения энергии.	4 балла
Указано, что часть взаимодействий частицы с электронами является упругой, а часть — неупругой.	2 балла
Верно найдена максимальная сила хотя бы в одном из отдельных взаимодействий.	2 балла
Дан правильный ответ на первый вопрос задачи.	4 балла
Дан правильный ответ на второй вопрос задачи.	4 балла
Дан правильный ответ на третий вопрос задачи.	4 балла
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

### *Задача III.1.3.4. Электрические цепи (20 баллов)*

Одной из перспективных нанотехнологических разработок является мемристор — электрический элемент, сопротивление  $R$  которого зависит от полного прошедшего через него заряда  $q$ . В некоторой лаборатории изготовлен мемристор с зависимостью  $R(q) = \alpha q$ , где  $\alpha = 100 \text{ Ом/Кл.}$

1. Какое количество теплоты  $Q$  выделится в мемристоре если подключить его к источнику постоянного тока  $I = 0,2 \text{ А}$  за  $\tau = 10 \text{ с?}$
2. По завершении первых  $\tau$  с напряжение на контактах источника достигло верхнего ограничения и источник перешел в режим поддержания этого постоянного напряжения. Чему будет равно это напряжение?
3. Какой заряд пройдет через мемристор в этом режиме когда на мемристоре выделится еще  $Q$  Дж тепла?

### *Решение*

В согласии с законом Ома на участке цепи, напряжение на мемристоре будет равно произведению его сопротивления на протекающую через него силу тока:

$$U(q) = IR(q) = I\alpha q.$$

Следовательно, работа  $A$ , совершаемая электрическим полем при переносе малого заряда  $\Delta q$  через мемристор, равна:

$$A(q) = U(q)\Delta q = I\alpha q \Delta q.$$

Выделившаяся в проводнике теплота равна полной работе, соответствующей переносу заряда  $q$ , и должна быть, таким образом, расчитана как работа переменной силы, линейно зависящей от величины этого заряда. Сделать это проще всего, найдя площадь под графиком  $U(q)$ :

$$Q = \frac{I\alpha q^2}{2}.$$

При этом заряд  $q$ , который протечет через мемристор за время  $\tau$  при постоянной силе тока  $I$  равен:

$$q = I\tau \Rightarrow Q = \frac{I^3 \alpha \tau^2}{2} = 40 \text{ Дж.}$$

После перехода источника в режим постоянного напряжения, выделяющееся тепло может быть расчитано по более простой формуле:

$$Q = Uq',$$

где  $q'$  — прошедший на этом этапе заряд.

При этом напряжение будет постоянно поддерживаться на максимальной для прошлого этапа величине:

$$U = I\alpha q = I^2 \alpha \tau = 40 \text{ В.}$$

Приравнивая выделенные при постоянном токе и при постоянном напряжении количества теплоты, получим

$$\cancel{I^2 \alpha \tau q'} = \frac{I^3 \alpha \tau^2}{2} \Rightarrow q' = \frac{I\tau^2}{2} = 1 \text{ Кл.}$$

**Ответ:**

1.  $Q = \frac{I^3 \alpha \tau^2}{2} = 40 \text{ Дж.}$
2.  $U = I\alpha q = I^2 \alpha \tau = 40 \text{ В.}$
3.  $q' = \frac{I\tau}{2} = 1 \text{ Кл.}$

### *Критерии оценивания*

Верно записан закон Ома для участка цепи или закон Джоуля-Ленца.	3 балла
Показано, что напряжение или мощность в цепи на первом этапе будет линейно зависеть от прошедшего заряда.	2 балла
Показано, что общая выделившаяся теплота является результатом действия переменной силы (движения заряда через переменное напряжение) и должна быть вычислена как площадь под графиком (из среднего значения напряжения).	3 балла
Показана связь прошедшего заряда и силы тока в цепи.	2 балла
Дан правильный ответ на первый вопрос задачи.	4 балла
Дан правильный ответ на второй вопрос задачи.	3 балла
Дан правильный ответ на третий вопрос задачи.	3 балла
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

### *Задача III.1.3.5. Динамика (20 баллов)*

Алгоритм автомобильного автопилота должен входить в поворот по радиусу, на  $\eta = 25\%$  превышающему расчетный минимально допустимый, для избежания заноса. Чтобы динамически отслеживать изменяющиеся дорожные условия, автопилот совершает крохотные попытки торможения, оценивая на их основании коэффициент сцепления с дорогой. Во время последнего такого торможения автомобиль, заблокировав колеса, сбросил скорость с  $v_1 = 15,1 \text{ м/с}$  до  $v_2 = 15 \text{ м/с}$ , пройдя расстояние  $l = 0,4 \text{ м}$ . Считать ускорение свободного падения  $g \approx 9,8 \text{ м/с}^2$ .

1. Какой коэффициент трения  $\mu$  вычислил автомобиль?
2. По какому радиусу автопилот должен повернуть автомобиль, продолжая двигаться со скоростью  $v_2$ ?
3. Геометрия дороги не позволяет автомобилю войти в следующий поворот по радиусу более  $R = 40 \text{ м}$ . До какого максимального значения  $v_3$  автопилоту следует сбросить скорость перед этим поворотом, чтобы продолжать следовать своему алгоритму?

Все движение автомобиля происходит на ровной горизонтальной дороге.

### *Решение*

Максимальная сила трения, создаваемая автомобилем при коэффициенте трения (сцепления)  $\mu$  на горизонтальной дороге, равна:

$$F = \mu N = \mu mg.$$

Работа этой силы на расстоянии  $l$  равна изменению кинетической энергии автомобиля:

$$\mu mgl = \frac{\mu}{2}(v_1^2 - v_2^2).$$

Таким образом, максимальное ускорение, которое могут обеспечить шины автомобиля, равно:

$$a = \mu g = \frac{v_1^2 - v_2^2}{2l} \Rightarrow \mu = \frac{v_1^2 - v_2^2}{2lg} \approx 0,38.$$

В повороте это ускорение играет роль центробежного:

$$\frac{v_1^2 - v_2^2}{2l} = \frac{v_2^2}{r_{min}},$$

где  $r_{min}$  — минимальный расчетный радиус.

Выражая отсюда  $r_{min}$  и умножая его на указанный в условиях запас  $1 + \eta$ , получим:

$$r = (1 + \eta) \frac{2v_2^2}{v_1^2 - v_2^2} l \approx 75 \text{ м.}$$

Для решения последнего пункта выразим требуемый минимально допустимый радиус  $R_{min}$  через известный возможный:

$$R_{min} = \frac{R}{1 + \eta}.$$

Подставим это значение в выражение для центробежного ускорения:

$$\frac{v_1^2 - v_2^2}{2l} = \frac{v_3^2(1 + \eta)}{R}.$$

Из этого уравнения легко выразить  $v_3$ :

$$v_3 = \sqrt{\frac{(v_1^2 - v_2^2)R}{2(1 + \eta)l}} \approx 11 \text{ м/с.}$$

**Ответ:**

1.  $\mu = \frac{v_1^2 - v_2^2}{2lg} \approx 0,38.$
2.  $r = (1 + \eta) \frac{2v_2^2}{v_1^2 - v_2^2} l \approx 75 \text{ м.}$
3.  $v_3 = \sqrt{\frac{(v_1^2 - v_2^2)R}{2(1 + \eta)l}} \approx 11 \text{ м/с.}$

### *Критерии оценивания*

Верно записана связь работы с изменением кинетической энергии или аналогичное кинематическое соотношение.	3 балла
Верно записано выражение для центростремительного ускорения.	2 балла
Найдена скорость, на которой автомобиль должен войти в последний поворот.	3 балла
Дан правильный ответ на первый вопрос задачи.	3 балла
Дан правильный ответ на второй вопрос задачи.	4 балла
Дан правильный ответ на третий вопрос задачи.	5 баллов
Всего:	20 баллов
Отсутствует решение в общем виде.	-2 балла за пункт
Решение в общем виде доведено не до конца.	-1 балл за пункт
Отсутствуют необходимые единицы измерения в ответе.	-1 балл за пункт
Незначительная ошибка в алгебраических преобразованиях.	-2 балла за пункт
Арифметическая ошибка в ответе при верном ответе в общем виде.	-1 балла за пункт
Ответ дан только в общем виде, число отсутствует.	-1 балл за пункт
Явно небрежное оформление при верном ходе решения.	-1 балл за пункт

# Командный практический тур

На заключительном этапе участникам необходимо создать и запустить автоматизированную транспортную систему для доставки груза до адресата без вмешательства человека. Система состоит из трех устройств: беспилотного автомобиля, распределительного хаба и квадрокоптера.

Все перечисленные устройства работают на полигоне городской среды с дорогами, перекрестками, дорожными знаками, пешеходами и светофорами, зданиями.

Беспилотный автомобиль доставляет груз к распределительному хабу через весь город, соблюдая правила дорожного движения, реагируя на пешеходов и светофоры. Распределительный хаб оснащен мостовым краном, который собирает и сортирует грузы, привезенные автомобилем. Он выбирает определенный груз из имеющихся и передает его в захват квадрокоптера. Квадрокоптер располагается на крыше сортировочного хаба, он захватывает груз и доставляет его в центр разгрузочной площадки, располагающейся на одной из крыш зданий полигона, после чего совершает приземление на специально оборудованную крышу.

Решения участников проверяются на полигоне.

## Требования к команде

Команде предстоит освоить работу с компьютерным зрением и нейронными сетями в различных сферах робототехники: беспилотный автомобиль на базе компьютерного зрения, квадрокоптер в режиме автономного полета, автоматизированный мостовой кран сортировочного хаба.

**Количество участников в команде:** 3–4. Состав команды приведен ниже.

- **Программист беспилотного автомобиля.** Программирование на Python, работа с компьютерным зрением и нейронными сетями в задачах беспилотного автомобиля, передвигающегося в условиях воссозданной городской среды.
- **Программист квадрокоптера.** Программирование на Python, работа с компьютерным зрением и нейронными сетями в задачах квадрокоптеров, осуществляющих навигацию над полигоном воссозданной городской среды.
- **Программист распределительного хаба.** Программирование на Python, работа с компьютерным зрением и нейронными сетями для решения задачи расшифровки цветных меток и сортировки грузов.
- **Капитан команды.** Программирование на Python, распределение задач по участникам команды, отслеживание дедлайнов, работа с компьютерным зрением и нейронными сетями.

## Оборудование и программное обеспечение

### Требования к рабочему месту и ПО

Рабочее место участников:

- Компьютер с выходом в сеть интернет.
- Наушники и микрофон для голосовой связи с оператором.
- Интернет канал — 8 мегабит/с.
- ПО для коммуникации: Telegram, Discord, Zoom.
- ПО для связи с операторами оборудования: AnyDesk.
- Любая среда программирования для Python.
- Браузер для просмотра видеотрансляций с полигона.

## Полигон

На полигоне присутствуют 4 объекта:

1. Беспилотный автомобиль «Айкар-1»;
2. Беспилотный автомобиль «Айкар-2»;
3. Квадрокоптер «Пионер Макс»;
4. Распределительный хаб.

Каждым объектом управляет оператор — посредник между участниками и устройством.

В строго отведенное время (см. расписание) участники подключаются к компьютерам операторов при помощи ПО AnyDesk, голосовая связь поддерживается через Discord. Участники присыпают оператору файлы с написанными программами. По указанию участников оператор выполняет следующие действия:

- загружает программы на устройство и запускает их;
- переносит устройство в разные положения на полигоне;
- редактирует код программ.

Участники могут самостоятельно редактировать программы на компьютере оператора и самостоятельно загружать программы на устройство.

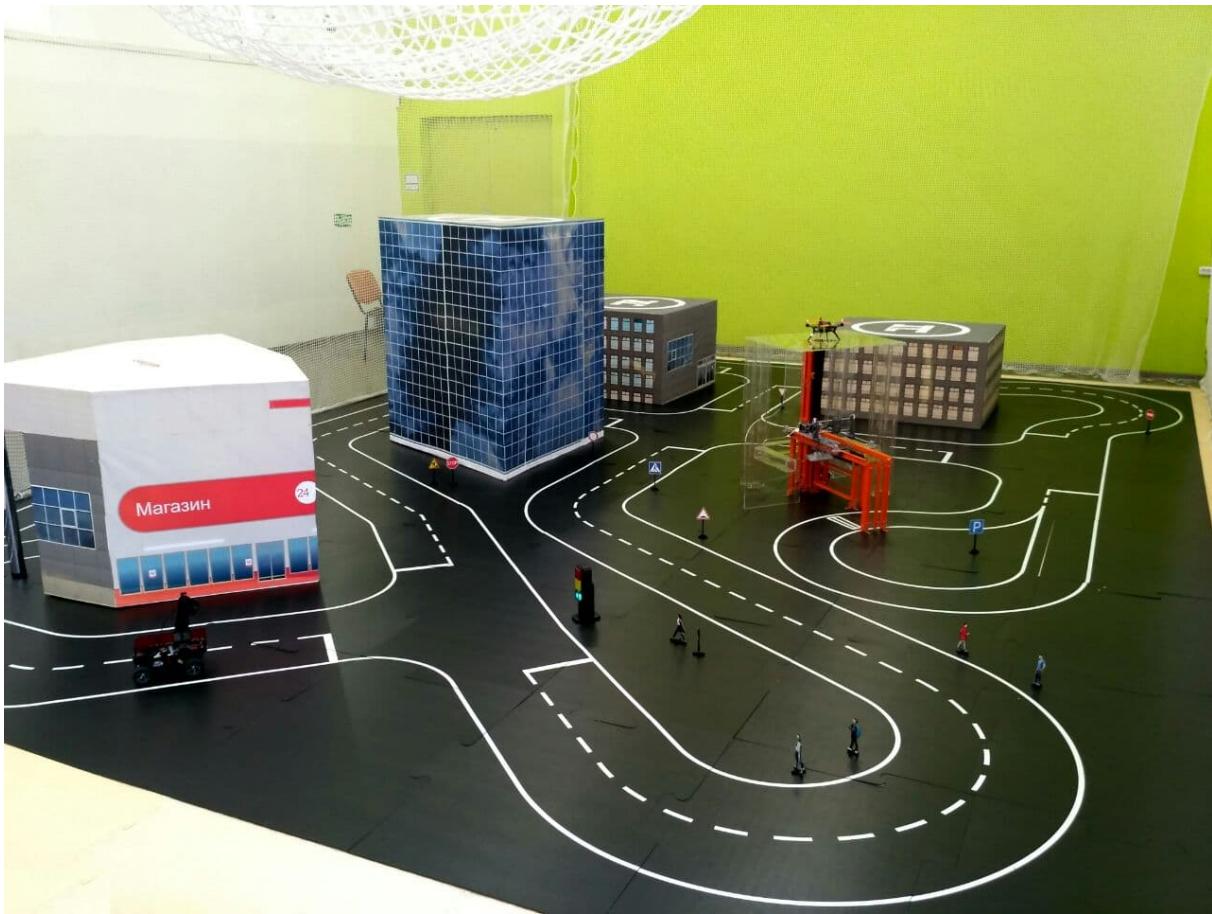
Для выполнения задачи с квадрокоптером каждой команде дается свой испытательный стенд (моделирует квадрокоптер). К стенду можно подключаться и проводить испытания без участия оператора. При необходимости осуществления полета квадрокоптера оператор переносит SD-карту стенда в квадрокоптер.

Результат работы устройства и все происходящее на полигоне участники могут наблюдать через трансляции в YouTube с разных камер.

Во время между подключениями к устройствам участники работают на своих рабочих местах: создают и редактируют программы.

## Описание задачи

Три устройства: беспилотный автомобиль, распределительный хаб и квадрокоптер — работают на полигоне городской среды с дорогами, перекрестками, дорожными знаками, пешеходами и светофорами, зданиями.



Беспилотный автомобиль доставляет груз к распределительному хабу через весь город, соблюдая правила дорожного движения, реагируя на пешеходов и светофоры. Распределительный хаб оснащен мостовым краном, который собирает и сортирует грузы, привезенные автомобилем. Он выбирает определенный груз из имеющихся и передает его в захват квадрокоптера. Квадрокоптер располагается на крыше сортировочного хаба, он захватывает груз и доставляет его в центр разгрузочной площадки, расположенной на одной из крыш зданий полигона, после чего совершает приземление на специально оборудованную крышу.

Устройства, запрограммированные участниками, должны поочередно выполнить свою часть задачи и реализовать доставку груза до адресата без участия человека.

Задача разбита на несколько более простых подзадач. В течение трех дней участники решают эти подзадачи и получают за них баллы. Четвертый день — день финальных заездов, когда участники должны продемонстрировать работу всей транспортной системы в целом. В случае, если система доставки груза полностью работоспособна, участники получают баллы.

### ***Подзадачи беспилотного автомобиля***

Программы для управления беспилотным автомобилем Айкар и их описание будет предоставлено участникам 14 марта в 13:00 (МСК).

Участникам были представлены программы для управления моделью беспилотного автомобиля Айкар. Они приложены в папке «Описание программ для управления Айкаром». В описаниях содержится задача, за которую не начислялись баллы, но без ее решения невозможно решить следующие задачи.

### *Задача III.2.3.1. Задача по программированию VOSTOK UNO*

Плата контроллер модели беспилотного автомобиля — VOSTOK UNO.

VOSTOK UNO выполняет программу «Kontroller.cpp». Программа должны принимать данные от Raspberry Pi и выдавать соответствующие им сигналы для сервоприводов и двигателя. Программа «Kontroller.cpp» не дописана до конца — отсутствует прием сообщений от Raspberry. Ознакомьтесь с кодом программы и доработайте его. Raspberry Pi и VOSTOK UNO обмениваются данными через UART. VOSTOK UNO использует Serial1.

Синтаксис написания программы для VOSTOK UNO совпадает с синтаксисом Arduino. Вы можете редактировать программу в Arduino IDE и прислать .ino файл. Не подключайте библиотеки сверх тех, которые уже подключены. Все пользовательские функции должны быть объявлены до того как будут использованы.

#### *1. Проехать 3 метра и остановиться (2 балла)*

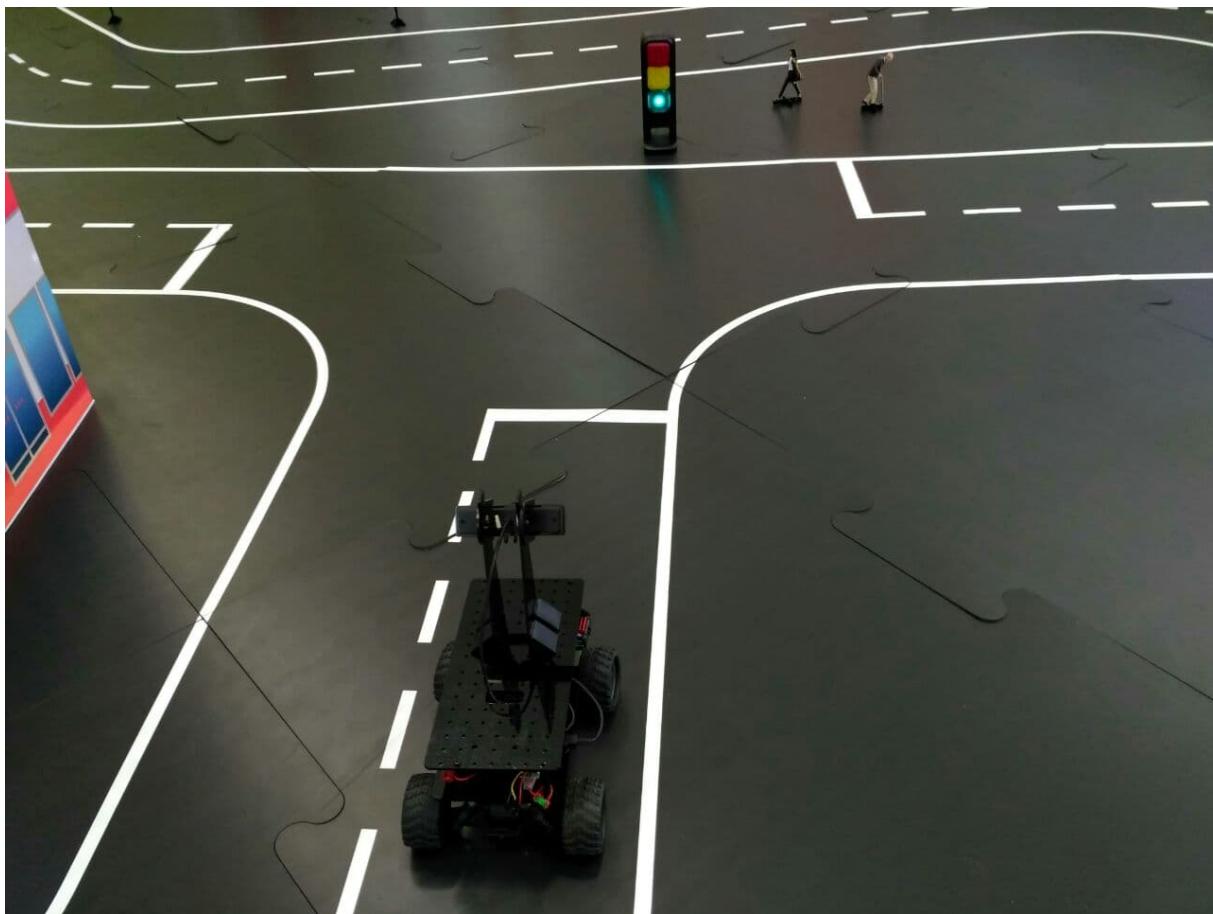
Айкар ставится в одну из трех стартовых позиций, позиция выбирается случайно.



Задание считается выполненным, если беспилотный автомобиль остановился проехав  $3 \text{ м} \pm 10 \text{ см}$ . Для проверки проводится 3 заезда, каждый должен быть успешным. Успешные попытки должны идти подряд.

#### *2. Определить длительность горения сигналов светофора (5 баллов)*

Айкар устанавливается на расстоянии от 10 до 50 см до перекрестка со светофором:



Светофор работает в автоматическом режиме. Стартовый сигнал светофора не известен. Всего сигналов пять: красный, желтый, зеленый, красный + желтый, мигающий зеленый.

Задача считается выполненной, если через два цикла работы светофора программа беспилотника вывела длительности горения сигналов светофора с точностью до 0,5 с.

Для проверки проводятся 2 испытания, каждое должно пройти успешно. Успешные попытки должны идти подряд.

### ***3. Стартовать на зеленый сигнал светофора и проехать 4 м (3 балла)***

Айкар ставится в одну из трех стартовых позиций, позиция выбирается случайно. На обочине дороги на расстоянии от 20 до 30 см от Айкара устанавливается светофор.



Светофор работает в автоматическом режиме. Стартовый сигнал светофора не известен. Всего сигналов пять: красный, желтый, зеленый, красный + желтый, мигающий зеленый.

Задача считается выполненной, если Айкар начал движение во время зеленого сигнала светофора, проехал  $4\text{ м} \pm 10\text{ см}$  и остановился.

Для проверки проводятся 2 заезда, каждый должен быть успешным. Успешные попытки должны идти подряд.

#### *4. Остановитесь перед пешеходом (9 баллов)*

Айкар ставится в одну из трех стартовых позиций, позиция выбирается случайно. На расстоянии от 1,5 до 2 м от Айкара на полосу движения устанавливается пешеход.



Задание считается выполненным, если Айкар начал движение и остановился в  $30\text{--}50\text{ см}$  перед пешеходом. После исчезновения пешехода Айкар должен продолжить движение. Для проверки проводятся 2 заезда, каждый должен быть успешным. Успешные попытки должны идти подряд.

**5. Доехать до распределительного хаба и разгрузиться в отведенной для этого области (8 баллов)**

Айкар устанавливается в стартовое положение.



Задача считается выполненной, если Айкар доехал до распределительного хаба и оставил груз в зоне разгрузки. Для проверки проводятся 2 заезда, каждый должен быть успешным. Успешные попытки должны идти подряд.

**6. Довезти груз до распределительного хаба с учетом всех препятствий (13 баллов)**

Айкар устанавливается в стартовое положение.



Перед Айкаром располагается светофор и пешеход, также как в предыдущих подзадачах. Необходимо стартовать на зеленый сигнал светофора, остановиться перед пешеходом и продолжить движение, когда пешеход исчезнет.

Задача считается выполненной, если Айкар доехал до распределительного-хаба и оставил груз в зоне разгрузки. При этом стартовал на зеленый сигнал светофора и остановился перед пешеходом. Для проверки проводятся 2 заезда, каждый должен быть успешным. Успешные попытки должны идти подряд.

## Решение

```
#include <Arduino.h> // # это строка нужна только на VOSTOK-UNO
#include <Servo.h> // # подключение библиотеки для управления сервоприводами
Servo ESC; // # создание объекта связанного с драйвером двигателя
Servo SRV; // # создание объекта связанного с рулевым сервоприводом
Servo CARGO; // # создание объекта связанного с сервоприводом кузова
int Speed, Angle;
int width impulse(int angle){
    return angle * 11.1 + 500; // Пересчет градусов в длину импульса для сервопривода
}
void dropCargo(){ // Опрокидывает кузов и возвращает его в исходное положение через
→ 1,5 секунды
    setSpeed_(1500); // останавливаемся! опрокидывать кузов на ходу - плохая идея
    CARGO.write(90); // принять положение 90 градусов, для сервопривода кузова // кузов
→ опрокинут
    delay(1500); //
    CARGO.write(160); // принять положение 160 градусов, для сервопривода кузова //
→ кузов не опрокинут
}
void setAngle(int angle){ // повернуть рулевые колеса в положение соответствующее углу,
                           // преданному в качестве аргумента
    if(angle > 110){angle = 110;}
    if (angle < 70){angle = 70;}
    SRV.write(angle);
}
void setSpeed(int _speed){ // ехать с новой скоростью, значение скорости в аргументе
    // Speed = _speed;
    ESC.writeMicroseconds(_speed);
}
void setup() { // Задать настройки для приема сообщений от Raspberry Pi через UART
    Serial1.begin(9600); // назначаем скорость общения
    ESC.attach(9); // указываем pin драйвера двигателя
    SRV.attach(8); // указываем pin рулевого сервопривода
    CARGO.attach(10); // указываем pin сервопривода кузова
    ESC.writeMicroseconds(1500); // отправляем стартовый сигнал драйверу двигателя, он
→ запомнит его как сигнал "СТОП"
    CARGO.write(160); // возвращаем сервопривод кузова в положение "кузов не опрокинут"
    SRV.write(90); // устанавливаем рулевые колеса прямо
    delay(2000); // ждем 2с, чтобы драйвер двигателя "уловил" сигнал "СТОП"
}
void loop() {
    if(Serial1.available() > 0){
        String data = Serial1.readStringUntil('\n'); // читаем символы, пока не прочитаем
→ символ конца строки
        // TODO: надо распарсить полученную строку и понять, что нам передали: угол,
→ скорость или команду "опрокинуть кузов"
        // Допишите сюда код
        setSpeed(Speed); // выполняем если получили значение скорости
        setAngle(Angle); // выполняем если получили угол поворота
    }
}
```

```

        dropCargo(); // выполняем если получили команду "опрокинуть кузов"
    }
}
void loop() {
    if(Serial1.available() > 0){
        String data = Serial1.readStringUntil('\n'); // читаем символы, пока не прочитаем
        → символ конца строки

```

**Строки, которые нужно было дописать:**

```

if(data.startsWith("SPEED")){
    //если сообщение начинается с "SPEED"
    String speed_str = data.substring(6); //оставляем символы идущие после первых
    → шести
    Speed = speed_str.toInt(); // переводим число следующее за "SPEED" в int
    setSpeed_(Speed); // задаем новое значение скорости
    Serial1.println("New speed: " + speed_str);

} else if(data.startsWith("ANGLE")){
    //если сообщение начинается с "ANGLE"
    String angle_str = data.substring(6); //оставляем символы идущие после первых
    → шести
    Angle = angle_str.toInt(); // переводим число следующее за "ANGLE" в int
    setAngle(Angle); // задаем новый угол поворота рулевых колес
    Serial1.println("New angle: " + angle_str);

} else if(data.startsWith("CARGO")){
    //если сообщение начинается с "CARGO"
    dropCargo(); //опрокидываем кузов
    Serial1.println("Cargo was dropped!");

} else { //иначе считаем, что пришла неопознанная команда
    Serial1.println("Unknown command: " + data);
}
}
}

```

### 1. Проехать 3 метра и остановиться

Задачу можно было решить двумя методами. Первый — нестабильный необходимо было отмерять время движения с помощью `time.monotonic`, и когда время превышает некоторые значение останавливать модель беспилотного автомобиля. Время и скорость модели, необходимо подбирать экспериментально.



Второй вариант решения задачи — стабильный. На участке трассы, где полагалось проехать три метра, на правой полосе, через каждые 25 см были нанесены метки.

Подсчет меток позволял точно определить пройденное расстояние. Ниже представлен основной цикл обработки кадров, в нем проводится подсчет меток.

```

counter = 0
flag = 0
while key != ESCAPE:
    status, frame = beholder_client.get_frame(0.25) # читаем кадр из очереди
    if status == beholder.Status.OK: # Если кадр прочитан успешно ...
        cv2.imshow("Frame", frame) # выводим его на экран
        img = cv2.resize(frame, SIZE)
        binary = binarize(img, d=1) # бинаризуем изображение
        perspective = trans_perspective(binary, TRAP, RECT, SIZE)
        # извлекаем область изображения перед колесами автомобиля
        left, right = centre_mass(perspective, d=1) # находим левую и правую линии
        → разметки
        err = 0 - ((left + right) // 2 - SIZE[0]//2) # вычисляем отклонение середины
        → дороги от центра кадра
        if abs(right - left) < 100:
            err = last
            print("LAST")
        angle = int(90 + KP * err + KD * (err - last)) # Вычисляем угол поворота колес
        if angle < 72:
            angle = 72
        elif angle > 108:
            angle = 108
        last = err
        # set_speed(speed)
        set_angle(angle)
        key = cv2.waitKey(1)
        # считаем сумму значений пикселей в окне захвата и сравниваем с порогом
        # размер окна захвата и порога определяем экспериментально
        if numpy.sum(perspective[50:150, 250:]) >= porog:
            if flag == 0: # если метка попала в окно захвата первый раз - увеличиваем
                → счетчик
                counter += 1
            flag = 1 # меняем значение флага, чтобы не посчитать эту метку дважды
        else:
            flag = 0

        if counter >= 13: # если мы насчитали больше 13 меток - останавливаемся.
            set_speed(1500)
    elif status == beholder.Status.EOS: # Если сервер прервал передачу
        print("End of stream")
        break
    elif status == beholder.Status.Error: # Если кадр пришел поврежденным
        print("Error")
        break
    elif status == beholder.Status.Timeout: # Если очередь кадров пуста.
        # Do nothing
        pass

```

## 2. Определить длительность горения сигналов светофора

Для того чтобы определить длительности горения сигналов светофора, необходимо сначала его детектировать, а затем определить сигнал и время, в течении ко-

торого сигнала не изменялся. Самый надежный способ детектирования светофора — использовать нейросетевой детектор, например, *yolo-tiny*. При обучении детектора получаются файл с весовым коэффициентом и файл с конфигурацией модели нейронной сети. Ниже представлен код для загрузки детектора из файлов и получения данных о сдектектированных пешеходах.

```
# порог уверенности сети в детектировании объекта, все ниже отсекается
CONFIDENCE_THRESHOLD = 0.5
# IoU для подавления накладывающихся друг на друга рамок
NMS_THRESHOLD = 0.4
class_names = ["human", "trf_light"] # список имен классов для которых обучался
→ детектор
def load_model(name):
    # загружаем из файлов весовые коэффициенты и конфигурацию сети
    net = cv2.dnn.readNet("yolov4-tiny.weights", "yolov4-tiny.cfg")
    # в зависимости от видеокарты нужно переключать BACKEND
    net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
    net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
    # net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA_FP16)
    model = cv2.dnn_DetectionModel(net) # создаем объект связанный с загруженными нами
    → данными
    # указываем размер обрабатываемого сетью изображения и необходимость нормировать
    → значения пикселей на 255
    model.setInputParams(size=(416, 416), scale=1 / 255)
    return model
def detect_trf_lights(model, frame):
    classes, scores, boxes = model.detect(frame, CONFIDENCE_THRESHOLD, NMS_THRESHOLD)
    trf_lights_box = []
    for (classid, score, box) in zip(classes, scores, boxes):
        if class_names[classid[0]] == "trf_light":
            box = box.astype(np.int32)
            trf_lights_box.append([box[0], box[1], box[0]+box[2], box[1]+box[3]])
            #cv2.rectangle(cutted_frame, box, (0, 233, 233), 2)
    return trf_lights_box
```

Изображение сдектектированного светофора приводится к стандартному размеру и переводится в цветовое пространство HLS. Изображение делится на три части, соответствующие огням светофора. В каждой из областей суммируются составляющие пикселей по каналу S. Если значение для области выше экспериментально определяемого порога, то считаем — огонь горит. Осталось для каждого отличного от предыдущего состояния светофора засечь длительность.

```
#light - изображение сдектектированного светофора
#light - делим изображение на три части и считаем число светлых пикселей
r = np.count_nonzero(light[8:24])
y = np.count_nonzero(light[40:56])
g = np.count_nonzero(light[64:96])
#определяем какие огни горят
r = r > 200
y = y > 200
g = g > 20
#определяем сигнал светофора
if r and y:
    state = "yellow-red"
elif r:
    state = "red"
elif y:
```

```

state = "yellow"
elif g:
    state = "green"
elif not (r or y or g):
    state = "none"
else:
    state = "unknown"

```

### 3. Стартовать на зеленый сигнал светофора и проехать 4 м

Подсчет пройденного расстояния осуществляется так же как в предыдущей подзадаче. Таким образом, решение задачи сводится к детектированию зеленого сигнала светофора. Задачу можно решить несколькими способами: можно детектировать единственно нужный зеленый сигнал светофора, а можно детектировать сам светофор и определять, какой сигнал он подает. Второй способ более сложен и избыточен для этой задачи.

Для детектирования зеленого сигнала светофора можно было воспользоваться алгоритмом детектирования объектов по цветам. Сначала бинаризовать изображение так, чтобы зеленые пиксели стали белыми, а все остальные черными. Посчитать число белых пикселей и по их количеству судить о наличии или отсутствии сигнала. Задачу усложняет то, что кроме зеленого сигнала светофора есть мигающий зеленый, на который нельзя стартовать. Для того чтобы отличить мигающий зеленый от зеленого, надо было засечь время горения зеленого сигнала. Если оно превышало полсекунды, значит сигнал зеленый. Ниже приведен фрагмент основного цикла обработки кадров, в который добавлен алгоритм определения зеленого сигнала светофора.

```

count=0
stop = False
while key != ESCAPE:
    status, frame = beholder_client.get_frame(0.25) # читаем кадр из очереди
    if status == beholder.Status.OK: # Если кадр прочитан успешно ...
        cv2.imshow("Frame", frame) # выводим его на экран
        img = cv2.resize(frame, SIZE)
        binary = binarize(img, d=1) # бинаризуем изображение
        perspective = trans_perspective(binary, TRAP, RECT, SIZE)
        # извлекаем область изображения перед колесами автомобиля

        speed = 1500
        # бинаризуем исходный кадр так, чтобы только зеленые пиксели стали белыми
        # маска для бинаризации подбирается экспериментально
        green = cv2.inRange(frame,(110,110,15),(230,230,30))
        # Если белых пикселей больше некоторого порога, определяемого экспериментально
        if sum(sum(green)) > 1000:
            # Считаем что детектировали зеленый сигнал и увеличиваем счетчик кадров с таким
            # сигналом
            count+=1
            if count>=20:
                # Если 20 кадров подряд горел зеленый, значит он не мигающий и можно
                # ехать
                gr = True
                print("GReeEn")
            else:
                # Если зеленый сигнал не обнаружен, обнуляем счетчик кадров
                count=0

```

```

if gr == True:
    speed = 1439

```

#### 4. Остановится перед пешеходом

Для стабильного решения задачи с пешеходом необходимо было использовать нейросетевой детектор. Детектор следовало обучить так, чтобы он определял пешеходов на приемлемом по условию задачи расстоянии. Это достигается созданием датасета с пешеходами только определенного размера. Наиболее удобный детектор для данной задачи — `yolo-tiny`. Его можно найти в открытом доступе, его можно обучить, используя открытые облачные сервисы, он выдает FPS, достаточный для быстрого реагирования на пешехода. При обучении детектора получаются файл с весовым коэффициентом и файл с конфигурацией модели нейронной сети. Ниже представлен код для загрузки детектора из файлов и получения данных о сдектектированных пешеходах.

```

# порог уверенности сети в детектировании объекта, все ниже отсекается
CONFIDENCE_THRESHOLD = 0.5
# IoU для подавления накладывающихся друг на друга рамок
NMS_THRESHOLD = 0.4
class_names = ["human", "trf_light"] # список имен классов для которых обучался
→ детектор
def load_model(name):
    # загружаем из файлов весовые коэффициенты и конфигурацию сети
    net = cv2.dnn.readNet("yolov4-tiny.weights", "yolov4-tiny.cfg")
    # в зависимости от видеокарты нужно переключать BACKEND
    net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
    net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
    # net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA_FP16)
    model = cv2.dnn_DetectionModel(net) # создаем объект связанный с загруженными нами
    → данными
    # указываем размер обрабатываемого сетью изображения и необходимость нормировать
    → значения пикселей на 255
    model.setInputParams(size=(416, 416), scale=1 / 255)
    return model
def detect_human(model, frame):
    # Получаем результаты детектирования от сети
    classes, scores, boxes = model.detect(frame, CONFIDENCE_THRESHOLD, NMS_THRESHOLD)
    human_box = []
    for (classid, score, box) in zip(classes, scores, boxes):
        # Перебираем все объекты и для объектов с именем human сохраняем ограничивающие
        → рамки
        if class_names[classid[0]] == "human":
            box = box.astype(np.int32)
            human_box.append([box[0], box[1], box[0]+box[2], box[1]+box[3]])
            #cv2.rectangle(cutted_frame, box, (0, 233, 233), 2)
    return human_box

```

Детектирование пешехода проводится в основном цикле обработки кадров. Пока есть хотя бы один детектированный пешеход — стоим на месте.

#### 5. Доехать до распределительного хаба и разгрузиться в отведенной для этого области

Чтобы добраться до распределительного хаба, беспилотнику необходимо преодолеть пару перекрестков и остановиться у зоны разгрузки. Для того что преодолеть

перекресток, необходимо повернуть рулевые колеса на определенный угол и продолжать движение определенное количество времени. Время засекается с помощью `time.monotonic()`. Угол колес и время поворота определяется экспериментально.

Переход от алгоритма движения в своей полосе к алгоритму движения через перекресток осуществляется при детектировании стоп-линии перед перекрестком. Детектировать стоп-линию можно, подсчитав сумму значений пикселей в каждой строке бинаризованного изображения.

Если сумма превышает определенный порог, то считаем, что стоп линия детектирована. Индекс строки позволит понять, насколько стоп-линия далеко от беспилотника, и остановиться именно на том расстоянии от стоп линии, на котором требуется. При остановке следует учитывать инерцию движения беспилотника. Для ее гашения, можно на несколько миллисекунд подавать напряжение на двигатель в режиме реверса.

```
ret, frame = cam.read()
binary = cv2.inRange(frame, (200, 200, 200), (250, 250, 250))
vertical_hist = np.sum(binary, axis=1)
ind_max_str = np.argmax(vertical_hist)
```

Остановка перед распределительным хабом совершается после детектирования З-ей стоп линии, которой обозначена зона разгрузки. Для того чтобы поточнее попасть кузовом в зону разгрузки после детектирования стоп-линий, можно сдать назад.

## *6. Довезти груз до распределительного хаба с учетом всех препятствий*

Задача содержит в себе предыдущие задачи, которые необходимо совместить в один алгоритм. Сначала дождаться зеленого сигнала светофора. Затем, перейти к алгоритму движения по трассе с учетом возможного появления пешехода. Далее преодолеть два перекрестка, остановиться у разгрузочной области и разгрузиться. Для решения задачи можно использовать как один детектор, так и два. Детектор светофоров требуется только в самом начале.

### ***Задача III.2.3.2. Подзадачи распределительного хаба***

Распределительный хаб оснащен мостовым краном с магнитным захватом. Участники могут управлять им программно. Программы для управления распределительным хабом и их описание будут предоставлены участникам 14 марта в 13:00 (МСК).

Распределительный хаб работает с грузами в форме куба. Ребро куба 2,5 см. На грани куба нанесены цветные метки, кодирующие номер груза. Примеры меток и их описание будут представлены 14 марта в 13:00 (МСК).

#### ***1. Забрать из зоны разгрузки указанный груз и передать его квадрокоптеру (7 баллов)***

Участникам сообщается номер груза, доставленного автомобилем. Задание считается выполненным, если мостовой кран захватил груз, доставленный автомобилем, опустил его в первый накопитель и доставил по ленте конвейера к захвату квадрокоптера.

На старте, в зоне разгрузки всегда два груза.

## *2. Забрать весь грузы из зоны разгрузки, доставленный автомобилем, передать квадрокоптеру, а остальные рассортировать по накопителям (13 баллов)*

Участникам сообщается число, соответствующее метке груза, доставленного автомобилем. Задание считается выполненным, если мостовой кран захватил груз, доставленный автомобилем, опустил его в первый накопитель и передал по ленте конвейера к захвату квадрокоптера. Все остальные грузы должны быть разложены по второму, третьему и четвертому накопителю в порядке возрастания номеров.

На страте, в зоне разгрузки всегда четыре груза.

### *Решение*

#### *1. Забрать из зоны разгрузки указанный груз и передать его квадрокоптеру*

Задача сводится к тому, чтобы распознать груз с заданным номером. Эту задачу можно решать множеством разных способов. Один из способов — определить номера всех грузов и выбрать из них тот, который нам нужен.

Сначала необходимо детектировать грузы, затем провести с верхними гранями грузов пространственные преобразования, растянув грузы до квадрата. Определить четыре цвета, соответствующих цветовой маркировке грузов. Перевести цвета в цифры и определить, какая из четырех цифр первая.

Для решения поставленной задачи нам потребуются вспомогательные функции.

```
# поиск расстояния между двумя точками на плоскости
def find_distance_2(pts1, pts2):
    return math.sqrt(abs(pts1[0] - pts2[0]) ** 2 + abs(pts1[1] - pts2[1]) ** 2)
# поиск расстояния между двумя точками в трехмерном пространстве
def find_distance_3(pts1, pts2):
    return math.sqrt(abs(pts1[0] - pts2[0]) ** 2 + abs(pts1[1] - pts2[1]) ** 2 +
                    abs(pts1[2] - pts2[2]) ** 2)
def find_color(color):
    # позволяет найти ближайший к данному на вход, цвет из словаря
    # цвета воспринимаются как точки в трехмерном пространстве
    # возвращает номер цвета
    colors = {0: (255, 255, 255), # white
              1: (40, 40, 40), # black
              2: (95, 95, 230), # red
              3: (80, 160, 250), # orange
              4: (70, 240, 235), # yellow
              5: (120, 200, 0), # green
              6: (255, 205, 0), # cyan
              7: (210, 120, 10), # blue
              8: (225, 150, 90), # violet
              9: (205, 145, 215)} # magenta
    cur_col, minim = None, np.inf
    for num, col in colors.items():
        dist = find_distance_3(color, col)
        if dist < minim:
```

```

        minim = dist
        cur_col = num
    return cur_col
frame = cv2.resize(frame, (640, 480))
# обрезаем кадр так, чтобы остался только контейнер с грузами
frame = frame[int(frame.shape[0] * 0.17):-int(frame.shape[0] * 0.1),
              int(frame.shape[1] * 0.27):-int(frame.shape[1] * 0.35)]
# Применяем размытие для более однородных цветов грузов
blur = cv2.medianBlur(frame, 5)
canny = cv2.Canny(blur, 50, 250)
# Детектор Кенни находит прямые линии, находи контуры на изображении с линиями
contours = cv2.findContours(canny.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = contours[0] if len(contours) == 2 else contours[1]
# Перебираем контуры и проверяя их площадь и соотношение сторон.
# Если они подходят под квадрат - сохраняю их в список
bboxes, areas = [], []
for cnt in contours:
    rect = cv2.minAreaRect(cnt)
    box = cv2.boxPoints(rect)
    box = np.int0(box)
    area = int(rect[1][0] * rect[1][1])
    side1, side2 = find_distance_2(box[0], box[1]), find_distance_2(box[1], box[2])
    if area > 2500 and side1 * 0.5 < side2 < side1 * 1.5:
        bboxes.append(box)
        areas.append(area)
masks = []
# Создаем список масок - соответствующих выбранным контуром
for i in range(len(bboxes)):
    mask = np.zeros((frame.shape[0], frame.shape[1], 1))
    cv2.drawContours(mask, [bboxes[i]], -1, 1, -1, cv2.LINE_AA)
    masks.append(mask)
# Сравниваем друг с другом все маски и оставляем только самые крупные,
# на которые не накладываются более крупные маски
bad = []
for i in range(len(bboxes)):
    for j in range(i + 1, len(bboxes)):
        if np.any(cv2.bitwise_and(masks[i], masks[j])) and np.sum(masks[i] *
        ~ masks[j]) /
            np.sum(masks[i]) > 0.2:
            if areas[i] < areas[j]:
                bad.append(j)
            else:
                bad.append(i)
    for j in range(len(main_masks)):
        if np.sum(main_masks[j] * masks[i]) / np.sum(main_masks[j]) > 0.2:
            bad.append(i)
            break
    if i not in bad:
        main_masks.append(masks[i])
        main_bboxes.append(bboxes[i])
else:
    break
# Преобразуем части исходного изображения закрывающиеся масками в квадраты
for k, bbox in enumerate(main_bboxes):
    dst = np.float32([(0, 0), (0, 200), (200, 200), (200, 0)])
    transform = cv2.getPerspectiveTransform(bbox.astype(np.float32), dst)
    new_image = cv2.warpPerspective(frame.copy(), transform, (200, 200))
    # Для каждого полученного изображения груза, растянутого в квадрат,
    # находим цвета четырех точек, соответствующим центрам квадров цветовой маркировки
    # переводим найденные цвета в цифры, которые они кодируют

```

```

cur_colors = []
for m in range(2):
    for n in range(2):
        point = new_image[50 + n * 100, 50 + m * 100]
        cur_colors.append(find_color(point))

```

Таким образом, мы имеем координаты всех грузов и соответствующие им цифры. Осталось определить верный порядок расположения цифр.

```

for color in cur_color:
    if color[0] + color[1] == 11 and color[0] + color[2] != 11:
        final_color = color
    elif color[1] + color[3] == 11 and color[1] + color[0] != 11:
        final_color = [color[1], color[3], color[0], color[2]]
    elif color[3] + color[2] == 11 and color[3] + color[1] != 11:
        final_color = [color[3], color[2], color[1], color[0]]
    elif color[2] + color[0] == 11 and color[2] + color[3] != 11:
        final_color = [color[2], color[0], color[3], color[1]]
    else:
        final_color = color
final_color = ''.join(str(x) for x in final_color)
all_colors.append(final_color)
all_coords.append(coords)

```

Осталось определить, какой из грузов на изображении соответствует данному изначально номеру. Это делается сравнением двух чисел.

## *2. Забрать весь груз из зоны разгрузки, доставленный автомобилем, передать квадрокоптеру, а остальные рассортировать по накопителям*

В предыдущей задаче можно было детектировать не все грузы, а только один, номер которого нам дан изначально. Но мы детектировали все грузы. Данная задача предполагает детектирование всех грузов и определение номеров каждого из них. Алгоритм идентичен алгоритму, приведенному в качестве решения предыдущей задачи.

### *Задача III.2.3.3. Подзадачи квадрокоптера*

SDK для программирования квадрокоптера: [https://github.com/geoscan/geoscan\\_pioneer\\_max](https://github.com/geoscan/geoscan_pioneer_max).

1. Для разработки и отладки программного кода команды-участники подключаются дистанционно к стендам. Каждой команде предоставляется один стенд. Смена стендов на протяжении соревнования не допускается.
2. Всю работу по программированию и запуску алгоритмов распознавания и полета участники производят самостоятельно. Оператор производит установку квадрокоптера на стартовую позицию, замену аккумуляторных батарей, а также по запросу участников смену графических меток для тестирования функций распознавания.
3. Запуск программы на квадрокоптере команда-участник производит только после подтверждения запуска оператором! При нарушении данного условия команда дисквалифицируется за нарушение техники безопасности при эксплуатации беспилотного летательного аппарата. Данное требование не относится к запуску программы полета в симуляторе на стенде команды.

4. За сохранность программного кода (например, путем создания бэкапов на локальном компьютере) несут ответственность команды-участники.
5. Сбор и разметку датасетов команды-участники выполняют самостоятельно.

**1. Выполнить захват груза, включение подсветки, взлет и зависание (2 балла)**

Квадрокоптер располагается на крыше сортировочного хаба. Груз уже доставлен к захвату квадрокоптера. Задание считается выполненным, если квадрокоптер последовательно совершил следующие действия:

1. включение магнитного захвата;
2. взлет с крыши распределительного хаба на произвольную высоту вместе захваченным грузом;
3. включение светодиодной индикации (цвет — красный);
4. зависание на 10 с;
5. выключение светодиодной индикации.

**2. Найти и распознать логотип получателя груза на крыше №1 (11 баллов)**

Перед стартом участникам сообщают логотип получателя груза, находящийся на крыше №1. Квадрокоптер располагается на крыше сортировочного хаба. Груз уже доставлен к захвату квадрокоптера. Задание считается выполненным, если квадрокоптер последовательно совершил следующие действия:

1. действия первой подзадачи;
2. пролет до крыши здания №1;
3. детектирование логотипа получателя груза;
4. включение светодиодной индикации (цвет — зеленый);
5. отправление оператору текстового сообщения с названием верно определенного логотипа получателя груза;
6. зависание на 10 секунд;
7. выключение светодиодной индикации.

**3. Найти и распознать логотип получателя груза на одной из двух крыш (11 баллов)**

Перед стартом участникам сообщают логотип получателя груза, который может находиться на одной из двух крыш: №1 либо №2. Заранее неизвестно, на какой из двух крыш окажется логотип. Квадрокоптер располагается на крыше сортировочного хаба. Груз уже доставлен к захвату квадрокоптера. Задание считается выполненным, если квадрокоптер последовательно совершил следующие действия:

1. действия первой подзадачи;
2. пролет до крыши здания №1;

3. детектирование логотипа получателя груза на крыше №1 и обоснить запятыми поясняющее выражение на крыше №2;
4. включение светодиодной индикации (цвет — зеленый);
5. отправление оператору текстового сообщения с названием верно определенного логотипа получателя груза;
6. зависание на 10 секунд;
7. выключение светодиодной индикации.

Всегда необходимо начинать детектирование с крыши №1 и, только если на ней не оказалось логотипа получателя, совершать перелет к крыше №2.

#### ***4. Доставить груз в центр логотипа получателя (9 баллов)***

Перед стартом участникам сообщают логотип получателя груза, который может находиться на одной из двух крыш: №1 либо №2. Заранее неизвестно, на какой из двух крыш окажется логотип. Квадрокоптер располагается на крыше сортировочного хаба. Груз уже доставлен к захвату квадрокоптера. Задание считается выполненным, если квадрокоптер последовательно совершил следующие действия:

1. действия третьей подзадачи;
2. центрирование над логотипом получателя;
3. аккуратный сброс груза.

Всегда необходимо начинать детектирование с крыши №1 и, только если на ней не оказалось логотипа получателя, совершать перелет к крыше №2. Чем ближе груз окажется к центру логотипа, тем больше баллов получат участники. Груз обязательно должен осться на крыше здания. Близость груза к центру логотипа оценивается после полной остановки груза. Для наиболее точной доставки груза рекомендуется опуститься как можно ниже или приземлиться на логотип получателя и, только после этого отключать магнитный захват.

#### ***Критерии оценивания***

- 9 баллов — центр груза находится не далее 5 см от центра метки;
- 6 баллов — центр груза находится на далее 10 см от центра метки;
- 3 балла — центр груза находится на расстоянии более 10 см от центра метки;
- 0 баллов — груз упал с крыши.

#### ***5. Выполнить доставку груза и произвести посадку в центре посадочной площадки (7 баллов)***

Перед стартом участникам сообщают логотип получателя груза, который может находиться на одной из двух крыш: №1 либо №2. Заранее неизвестно, на какой из двух крыш окажется логотип. Квадрокоптер располагается на крыше сортировочного хаба. Груз уже доставлен к захвату квадрокоптера. Задание считается выполненным, если квадрокоптер последовательно совершил следующие действия:

1. действия четвертой подзадачи;
2. перелет к крыше №3;

3. включение светодиодной индикации «посадка» (цвет — мигающий красный с интервалом 1 с);
4. зависание на 5 секунд;
5. центрирование над посадочной площадкой;
6. снижение с приземлением;
7. отключение светодиодной индикации.

Чем ближе к центру посадочной площадки приземлится квадрокоптер, тем больше баллов вы получите. Центром квадрокоптера считается центр магнита.

### *Критерии оценивания*

- 5 баллов — центр квадрокоптера находится не далее 5 см от центра метки;
- 3 балла — центр квадрокоптера находится не далее 10 см от центра метки;
- 1 балл — центр квадрокоптера находится на расстоянии более 10 см от центра метки;
- 0 баллов — квадрокоптер упал с крыши;
- дополнительные баллы: до 2 баллов.

### *Решение*

#### *1. Выполнить захват груза, включение подсветки, взлет и зависание*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import rospy
from gs_flight import FlightController, CallbackEvent
from gs_board import BoardManager
from gs_module import CargoController
rospy.init_node("task1_node") # инициализируем ноду
run = True # переменная отвечающая за работу программы
def callback(event): # функция обработки событий Автопилота
    global ap
    global run
    global cargo
    event = event.data
    if event == CallbackEvent.ENGINES_STARTED: # блок обработки события запуска
        → двигателя
        ap.takeoff() # отдаем команду взлета
    elif event == CallbackEvent.TAKEOFF_COMPLETE: # блок обработки события завершения
        → взлета
        cargo.changeAllColor(255.0, 0.0, 0.0) # включаем светодиоды на магнитном
        → модуле, (255, 0, 0) - соответствует красному цвету в цветовой модели RGB
        rospy.sleep(10) # ожидание 10 секунды
        cargo.changeAllColor() # выключаем светодиоды, функция changeAllColor берет
        → аргументов соответствует команде changeAllColor(0, 0, 0)
        run = False # включаем программу
board = BoardManager() # создаем объект бортового менеджера
ap = FlightController(callback) # создаем объект управления полета
cargo = CargoController() # создаем объект управления магнитным модулем
once = False # переменная отвечающая за первое вхождение в начало программы
while not rospy.is_shutdown() and run:
    if board.runStatus() and not once: # проверка подключения RPi к Пионеру
        cargo.on() # включаем магнит
```

```

ap.preflight() # отдаём команду выполнения предстартовой подготовки
once = True
pass

```

## 2. Найти и распознать логотип получателя груза на крыше №1

Для успешного выполнения задачи необходимо было модернизировать функцию обработки событий автопилота `callback` из прошлой задачи так, чтобы после отработки блока `TAKEOFF_COMPLITE` квадрокоптер полетел на крышу первого здания. Для этого необходимо было прочитать файл `objects.json`, который был выдан участникам вместе с симулятором. Внутри него были описаны координаты зданий. Необходимо было считать координаты первого здания из этого файла, а потом найти координаты углов здания. Функция получения координат описана ниже.

```

def get_points(): # функция получения координат здания
    coordinates = []
    with open("objects.json", "r") as f: # открываем файл на чтение
        objects = json.load(f) # загружаем данные из json файла
        # 1 - индекс первого здания в массиве объектов в файле json
        x = objects[1]["position"]["x"] # считываем координату x
        y = objects[1]["position"]["y"] # считываем координату y
        scale_x = objects[1]["scale"]["x"] # считываем размер здания по координате x
        scale_y = objects[1]["scale"]["y"] # считываем размер здания по координате y
        scale_z = objects[1]["scale"]["z"] # считываем высоту здания
        coordinates.append([x - scale_x * 0.3, y - scale_y * 0.3, scale_z + 0.3]) # ← высчитываем координаты левого верхнего угла со смещением в центр
        coordinates.append([x - scale_x * 0.3, y + scale_y * 0.3, scale_z + 0.3]) # ← высчитываем координаты правого верхнего угла со смещением в центр
        coordinates.append([x + scale_x * 0.3, y + scale_y * 0.3, scale_z + 0.3]) # ← высчитываем координаты правого нижнего угла со смещением в центр
        coordinates.append([x + scale_x * 0.3, y - scale_y * 0.3, scale_z + 0.3]) # ← высчитываем координаты левого нижнего угла со смещением в центр
    return coordinates

```

Для детектирования меток необходимо было использовать нейросетевой детектор. Одним из способов решения является использование `Yolo-tiny`. Ниже приведена функция загрузки модели и детектирования.

```

# порог уверенности сети в детектировании объекта, все ниже отсекается
CONFIDENCE_THRESHOLD = 0.5
# IoU для подавления накладывающихся друг на друга рамок
NMS_THRESHOLD = 0.4
class_names = ["hight_voltage", "olympiad80", "peace", "radiation", "smile", "start",
               "yin_yang"] # список имен классов для которых обучался детектор
def load_model():
    # загружаем из файлов весовые коэффициенты и конфигурацию сети
    net = cv2.dnn.readNet("yolov4-tiny.weights", "yolov4-tiny.cfg")
    model = cv2.dnn_DetectionModel(net) # создаем объект связанный с загруженными нами
    # данными
    # указываем размер обрабатываемого сетью изображения и необходимость нормировать
    # значения пикселей на 255
    model.setInputParams(size=(640, 480), scale=1 / 255)
    return model
def detect_label(model, frame):
    global class_names
    # Получаем результаты детектирования от сети

```

```

classes, scores, _ = model.detect(frame, CONFIDENCE_THRESHOLD, NMS_THRESHOLD)
index = scores.index(max(scores)) # находим индекс класса с самой большой
→ вероятностью
return class_names[classes[index][0]] # возвращаем название метки

```

Для работы нейронной сети необходимо получить изображение с камеры через ROS. Для этого следует получить сообщения от топика /pioneer\_max\_camera/image\_raw и с помощью CvBridge преобразовать сообщение ROS типа sensor\_msgs/Image в изображение OpenCv. Ниже приведена функция получения и преобразования кадра.

```

import rospy, cv2
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
bridge = CvBridge() # объявляем конвертер типов
def get_img():
    global bridge
    data = rospy.wait_for_message("/pioneer_max_camera/image_raw/", Image) #
    → синхронное получение изображения из ROS
    frame = bridge.imgmsg_to_cv2(data, 'bgr8') # конвертация кадра из ROS в Cv2
    return frame

```

Название метки необходимо было отправить оператору. Для этого нужно создать издатель топика /label типа std\_msgs/String. После детектирования необходимо через созданный издатель отправить строку. Ниже приведен код инициализации издателя.

```

from rospy import Publisher
from std_msgs.msg import String
to_operator = Publisher('/label', String, queue_size = 10) # инициализация издателя

```

С учетом всех функций функция обработки событий Автопилота принимает вид:

```

NEED_MARK = "peace" # метка требуемая оператором
point_number = 0 # счетчик точек
model = load_model()
coordinates = get_points() # считываем координаты из файла
def callback(event): # функция обработки событий Автопилота
    global ap
    global run
    global cargo
    global coordinates
    global point_number
    global model
    global to_operator
    event = event.data
    if event == CallbackEvent.ENGINES_STARTED: # блок обработки события запуска
        → двигателя
        ap.takeoff() # отдаем команду взлета
    elif event == CallbackEvent.TAKEOFF_COMPLETE: # блок обработки события завершения
        → взлета
        cargo.changeAllColor(255.0, 0.0, 0.0) # включаем светодиоды на магнитном
        → модуле, (255, 0, 0) - соответствует красному цвету в цветовой модели RGB
        rospy.sleep(10) # ожидание 10 секунды
        cargo.changeAllColor() # выключаем светодиоды, функция changeAllColor берет
        → аргументов соответствует команде changeAllColor(0, 0, 0)

```

```

ap.goToLocalPoint(coordinates[point_number][0], coordinates[point_number][1],
                   ↵ coordinates[point_number][2])
elif event == CallbackEvent.POINT_REACHED:
    point_number += 1 # наращиваем счетчик точек
    image = get_img() # получаем изображение с камеры
    image = cv2.resize(image, (640, 480), interpolation = cv2.INTER_AREA) #
    ↵ изменяем размер изображения, для детектирования
    label = detect_label(model, image) # детектируем
    if label != NEED_MARK: # если детектирования метка не совпадает с необходимой,
    ↵ летим на следующую координату
        # если кол-во пройденных координат меньше общего кол-во точек летим на
        ↵ следующую точку, иначе прекращаем программу
        if point_number < len(coordinates): # если кол-во пройденных координат
        ↵ меньше общего кол-во точек летим на следующую точку
            ap.goToLocalPoint(coordinates[point_number][0],
                               ↵ coordinates[point_number][1], coordinates[point_number][2])
        else:
            run = False # прекращаем программу
    else:
        cargo.changeAllColor(0, 255, 0) # включаем светодиоды на магнитном модуле,
        ↵ (0, 255, 0) - соответствует зеленому цвету в цветовой модели RGB
        to_operator.publish(label) # публикуем название, чтобы оператор мог
        ↵ увидеть
        rospy.sleep(10) # зависание 10 секунд
        cargo.changeAllColor() # выключаем светодиоды, функция changeAllColor берет
        ↵ аргументов соответствует команде changeAllColor(0, 0, 0)
        run = False # прекращаем программу

```

### 3. Найти и распознать логотип получателя груза на одной из двух крыши

Для успешного выполнения задания необходимо модифицировать функцию чтения json файла так, чтобы в список координат добавились координаты второго здания. Код модифицированной функции приведен ниже.

```

def get_points():
    coordinates = []
    with open("objects.json", "r") as f: # открываем файл на чтение
        objects = json.load(f) # загружаем данные из json файла
        # 1 и 0 - индексы 1 и 2 здания соответственно в массиве объектов в файле json
        for index in [1, 0]:
            x = objects[index]["position"]["x"] # считываем координату x
            y = objects[index]["position"]["y"] # считываем координату y
            scale_x = objects[index]["scale"]["x"] # считываем размер здания по
            ↵ координате x
            scale_y = objects[index]["scale"]["y"] # считываем размер здания по
            ↵ координате y
            scale_z = objects[index]["scale"]["z"] # считываем высоту здания
            coordinates.append([x - scale_x * 0.3, y - scale_y * 0.3, scale_z + 0.3])
            ↵ # высчитываем координаты левого верхнего угла со смещением в центр
            coordinates.append([x - scale_x * 0.3, y + scale_y * 0.3, scale_z + 0.3])
            ↵ # высчитываем координаты правого верхнего угла со смещением в центр
            coordinates.append([x + scale_x * 0.3, y + scale_y * 0.3, scale_z + 0.3])
            ↵ # высчитываем координаты правого нижнего угла со смещением в центр
            coordinates.append([x + scale_x * 0.3, y - scale_y * 0.3, scale_z + 0.3])
            ↵ # высчитываем координаты левого нижнего угла со смещением в центр
    return coordinates

```

#### 4. Доставить груз в центр логотипа получателя

Для центрирования необходимо знать координаты центра метки, с этой целью нужно модифицировать функцию `detect_label` так, чтобы она помимо названия возвращала координаты центра метки.

```
def detect_label(model, frame):
    global class_names
    # Получаем результаты детектирования от сети
    classes, scores, boxes = model.detect(frame, CONFIDENCE_THRESHOLD, NMS_THRESHOLD)
    index = scores.index(max(scores)) # находим индекс класса с самой большой
    → вероятностью
    x = (boxes[index][0] + boxes[index][2]) / 2 # высчитываем центр метки
    y = (boxes[index][1] + boxes[index][3]) / 2 # высчитываем центр метки
    return class_names[classes[index][0]], x, y # возвращаем название и координаты
    → центра метки
```

Также для успешного центрирования необходимо рассчитать разницу между координатами центра камеры и центром метки, перевести ее в метры, чтобы на полученное расстояние выполнить смещение. Функция расчета координат приведена ниже.

```
from gs_navigation import NavigationManager
from math import tan, radians
def centering(x, y): # на вход получаем координаты центра метки
    horizontal_view = 62.2 # горизонтальный угол обзора в градусах PiCamera V2,
    → установленной на компьютере
    horizontal_view = radians(horizontal_view) # переводим градусы в радианы
    image_horizontal_size = 640 # горизонтальный размер изображения в пикселях
    copter_height = 0.3 # высота над зданием (задали при чтении json)
    # находим тангенс половины угла обзора, затем из определения тангенса в
    → прямоугольном треугольнике находим расстояние в сантиметрах от центра камеры
    → до края и делим на кол-во пикселей
    pixel_in_meter = tan(horizontal_view / 2) * 0.3 / (640 / 2)
    delta_x = (x - (650 / 2 - 1)) * pixel_in_meter # разница между координатами X
    → центра изображения и центром метки в метрах
    delta_y = (y - (480 / 2 - 1)) * pixel_in_meter # разница между координатами Y
    → центра изображения и центром метки в метрах
    copter_x, copter_y, copter_z = NavigationManager().lps.position() # получаем
    → текущие координаты компьютера
    # поскольку начало координат компьютера и координаты отсчета изображения
    → инвертированы относительно друг друга
    # прибавляем координаты X к Y, а Y к X
    copter_x += delta_y
    copter_y += delta_x
    copter_z -= 0.15 # уменьшаем высоту полета, чтобы более точно сбросить груз
    copter_x -= 0.05 # смещаем компьютер на расстояние от центра камеры до центра магнита
    → (расстояние приблизительное)
    return copter_x, copter_y, copter_z
```

Далее необходимо модифицировать функцию обработки событий автопилота так, чтобы после обнаружения метки и индикации выполнялось центрирование, а после него — сброс груза. Модифицированная функция `callback` приведена ниже.

```
with_cargo = True # с грузом ли компьютер
by_coord = True # флаг полета с грузом по точкам
```

```

def callback(event): # функция обработки событий Автопилота
    global ap
    global run
    global cargo
    global coordinates
    global point_number
    global model
    global to_operator
    global with_cargo
    global by_coord
    event = event.data
    if event == CallbackEvent.ENGINES_STARTED: # блок обработки события запуска
        ← движателя
        ap.takeoff() # отдаем команду взлета
    elif event == CallbackEvent.TAKEOFF_COMPLETE: # блок обработки события завершения
        ← взлета
        cargo.changeAllColor(255.0, 0.0, 0.0) # включаем светодиоды на магнитном
        ← модуле, (255, 0, 0) - соответствует красному цвету в цветовой модели RGB
        rospy.sleep(10) # ожидание 10 секунды
        cargo.changeAllColor() # выключаем светодиоды, функция changeAllColor берет
        ← аргументов соответствует команде changeAllColor(0, 0, 0)
        ap.goToLocalPoint(coordinates[point_number][0], coordinates[point_number][1],
        ← coordinates[point_number][2])
    elif event == CallbackEvent.POINT_REACHED:
        if with_cargo: # если груз захвачен, выполняем полет между зданиями, если груз
            ← отпущен прекращаем работу программы
            if by_coord: # если выполняется полет по координатам выполняем
                ← детектирование меток, если нет - центрирование и сброс груза
                point_number += 1 # наращиваем счетчик точек
                image = get_img() # получаем изображение с камеры
                image = cv2.resize(image, (640, 480), interpolation = cv2.INTER_AREA)
                ← # изменяем размер изображения, для детектирования
                label, x, y = detect_label(model, image) # детектируем
                if label != NEED_MARK: # если детектированная метка не совпадает с
                    ← необходимой, летим на следующую координату
                    # если кол-во пройденных координат меньше общего кол-во точек
                    ← летим на следующую точку, иначе прекращаем программу
                    if point_number < len(coordinates): # если кол-во пройденных
                        ← координат меньше общего кол-во точек летим на следующую точку
                        ap.goToLocalPoint(coordinates[point_number][0],
                        ← coordinates[point_number][1],
                        ← coordinates[point_number][2])
                    else:
                        run = False # прекращаем программу
                else:
                    run = True # продолжаем программу
            else:
                cargo.changeAllColor(0, 255, 0) # включаем светодиоды на магнитном
                ← модуле, (0, 255, 0) - соответствует зеленому цвету в цветовой
                ← модели RGB
                to_operator.publish(label) # публикуем название, чтобы оператор
                ← мог увидеть
                rospy.sleep(10) # ожидание 10 секунд
                cargo.changeAllColor() # выключаем светодиоды, функция
                ← changeAllColor берет аргументов соответствует команде
                ← changeAllColor(0, 0, 0)
                new_x, new_y, new_z = centering(x, y) # рассчитываем новые
                ← координаты
                by_coord = False # отключаем полет по точкам
                ap.goToLocalPoint(new_x, new_y, new_z)
            else:
                cargo.off() # выключаем магнит

```

```

        with_cargo = False # переключаем флаг груза
else:
    run = False # прекращаем программу

```

## 5. Выполнить доставку груза и произвести посадку в центре посадочной площадки

Для создания мигающей индикации, независимой от зависания, необходимо организовать поток. Функция, которая будет запущена в потоке, приведена ниже.

```

import time
def target(): # функция, которую будет выполнять поток
    global cargo
    while True:
        cargo.changeAllColor(255.0, 0.0, 0.0) # включаем красные светодиоды
        time.sleep(1) # останавливаем поток на 1 секунду
        cargo.changeAllColor() # выключаем светодиоды
        time.sleep(1) # останавливаем поток на 1 секунду

```

Создание и запуск потока.

```

from threading import Thread
indication = Thread(target=target) # создание потока
indication.start() # запуск потока

```

Далее необходимо модифицировать функцию `callback` так, чтобы после сброса груза добавились точки маршрута для перемещения до зоны сброса. Координаты точек предлагалось подобрать вручную, основываясь на координатах объектов на полигоне и тестовых полетах. Для успешного полета достаточно было трех точек: точка между краями первого и второго здания, точка с координатой  $Y$  предыдущей точки и координатой  $X$  центра места посадки, точка центра места посадки. Функция получения этих координат приведена ниже.

```

def add_landing_coordinates(): # функция добавления координат до места посадки
    global coordinates
    _, _, copter_z = NavigationManager().lps.position() # получаем текущие координаты
    # компьютера
    landing_coord = [6.68, 1.52, 1.1] # координаты места посадки

    first_point_x = (coordinates[3][0] - coordinates[6][0]) / 2 # координата X между
    # левым нижнем углом первого здания и правого нижнего угла второго здания
    first_point_y = (coordinates[3][1] - coordinates[6][1]) / 2 # координата Y между
    # левым нижнем углом первого здания и правого нижнего угла второго здания
    first_point_z = copter_z + 0.2 # взлетаем на 0.2 метра, чтобы не зацепится об
    # крышу здания при смещении
    second_point_x = landing_coord[0] # координата как у центра места посадки
    second_point_y = first_point_y # координата, как точки между зданиями
    second_point_z = landing_coord[2] # координата высота, как у места посадки
    coordinates.append([first_point_x, first_point_y, first_point_z])
    coordinates.append([second_point_x, second_point_y, second_point_z])
    coordinates.append(landing_coord)

```

Для успешной посадки необходимо было использовать функцию `ap.disarm()`, вместо встроенной функции `ap.landing()`. Функция `landing` сажает коптер не ровно по оси  $Z$ , поскольку воздушные потоки, создаваемые квадрокоптером под собой,

отражаются от поверхности над ним, таким образом сдувая коптер по оси  $X$  и  $Y$ . Функция `disarm` выключает движки, поэтому если ее выполнить на достаточно малом расстоянии от места посадки, коптер под силой собственной тяжести упадет практически ровно вдоль оси  $Z$ .

Модифицированная функция `callback` приведена ниже.

```
def callback(event): # функция обработки событий Автопилота
    global ap
    global run
    global cargo
    global coordinates
    global point_number
    global model
    global to_operator
    global with_cargo
    global by_coord
    event = event.data
    if event == CallbackEvent.ENGINES_STARTED: # блок обработки события запуска
        ← движателя
        ap.takeoff() # отдаем команду взлета
    elif event == CallbackEvent.TAKEOFF_COMPLETE: # блок обработки события завершения
        ← взлета
        cargo.changeAllColor(255.0, 0.0, 0.0) # включаем светодиоды на магнитном
        ← модуле, (255, 0, 0) - соответствует красному цвету в цветовой модели RGB
        rospy.sleep(10) # ожидание 10 секунды
        cargo.changeAllColor() # выключаем светодиоды, функция changeAllColor берет
        ← аргументов соответствует команде changeAllColor(0, 0, 0)
        ap.goToLocalPoint(coordinates[point_number][0], coordinates[point_number][1],
        ← coordinates[point_number][2])
    elif event == CallbackEvent.POINT_REACHED:
        if with_cargo: # если груз захвачен, выполняем полет между зданиями, если груз
        ← отпущен летим к зоне посадки
            if by_coord: # если выполняется полет по координатам выполняем
                ← детектирование меток, если нет - центрирование и сброс груза
                point_number += 1 # наращиваем счетчик точек
                image = get_img() # получаем изображение с камеры
                image = cv2.resize(image, (640, 480), interpolation = cv2.INTER_AREA)
                ← # изменяем размер изображения, для детектирования
                label, x, y = detect_label(model, image) # детектируем
                if label != NEED_MARK: # если детектирования метка не совпадает с
                    ← необходимой, летим на следующую координату
                    # если кол-во проходимых координат меньше общего кол-во точек
                    ← летим на следующую точку, иначе прекращаем программу
                    if point_number < len(coordinates): # если кол-во проходимых
                        ← координат меньше общего кол-во точек летим на следующую точку
                        ap.goToLocalPoint(coordinates[point_number][0],
                        ← coordinates[point_number][1],
                        ← coordinates[point_number][2])
                    else:
                        run = False # прекращаем программу
                else:
                    cargo.changeAllColor(0, 255, 0) # включаем светодиоды на магнитном
                    ← модуле, (0, 255, 0) - соответствует зеленому цвету в цветовой
                    ← модели RGB
                    to_operator.publish(label) # публикуем название, чтобы оператор
                    ← мог увидеть
                    rospy.sleep(10) # зависание 10 секунд
                    cargo.changeAllColor() # выключаем светодиоды, функция
                    ← changeAllColor берет аргументов соответствует команде
                    ← changeAllColor(0, 0, 0)
```

```

    new_x, new_y, new_z = centering(x, y) # рассчитываем новые
    ← координаты
    by_coord = False # отключаем полет по точкам
    ap.goToLocalPoint(new_x, new_y, new_z)

else:
    cargo.off() # выключаем магнит
    with_cargo = False # переключаем флаг груза
    point_number = len(coordinates) # устанавливаем счетчик на кол-во
    ← элементов в массиве координат
    add_landing_coordinates() # добавляем координаты
    ap.goToLocalPoint(coordinates[point_number] [0],
    ← coordinates[point_number] [1], coordinates[point_number] [2])

else:
    point_number += 1 # наращиваем счетчик точек
    if point_number < len(coordinates): # если кол-во пройденных координат
    ← меньше общего кол-во точек летим на следующую точку, иначе выполняем
    ← посадку
    ap.goToLocalPoint(coordinates[point_number] [0],
    ← coordinates[point_number] [1], coordinates[point_number] [2])
else:
    indication = Thread(target=target) # создание потока индикации
    indication.start() # запускаем индикацию
    rospy.sleep(5) # зависание
    cargo.changeAllColor() # выключаем светодиоды, функция changeAllColor
    ← бер аргументов соответствует команде changeAllColor(0, 0, 0)
    ap.disarm() # выключаем двигатели
    run = False # прекращаем программу

```

## Финальные заезды

В день финальных заездов задача участников — продемонстрировать работу всей транспортной системы в целом. Груз, с которым стартует беспилотный автомобиль, необходимо доставить получателю без участия человека. Если хотя бы одно из трех устройств не справилось с поставленной задачей, то вся система считается неработоспособной, и баллы за заезд не начисляются. На демонстрацию работы всей транспортной системы командедается 20 минут, количество попыток ограничено только временем.

Финальный заезд разбит на элементы. Есть элементы, обязательные для выполнения, а есть опциональные. За час до старта своего заезда команда должна заявить, какие именно элементы она будет выполнять. Если хотя бы один элемент из заявленных не выполнен, баллы за финальный заезд не начисляются.

Беспилотный автомобиль должен доставить груз до распределительного хаба и выгрузить его в отведенной для этого области. За старт на зеленый сигнал светофора участники могут получить дополнительные баллы. За остановку перед пешеходом и продолжение движения после его исчезновения участники могут получить дополнительные баллы.

Распределительный хаб должен захватить разгруженный беспилотником груз и передать его в захват квадрокоптера. За сбор остальных грузов в зоне разгрузки и их сортировку по накопителям участники могут получить дополнительные баллы.

Квадрокоптер должен доставить груз как можно ближе к центру логотипа получателя и приземлиться на посадочной площадке.

### *Критерии оценивания*

<b>Элементы финального заезда</b>	<b>Необходимость выполнения</b>	<b>Баллы</b>
Довезти груз до распределительного хаба и выгрузить его в специально отведенной области	Обязательно к выполнению	$8 \cdot 2 = 16$
Стартовать по зеленому сигналу светофора	Опционально	$3 \cdot 2 = 6$
Остановиться перед пешеходом	Опционально	$9 \cdot 2 = 18$
<b>Итого</b>		<b>40</b>
Забрать из зоны разгрузки указанный груз и передать его квадрокоптеру.	Обязательно к выполнению	7
Забрать оставшиеся грузы из зоны разгрузки и рассортировать их по накопителям	Опционально	13
<b>Итого</b>		<b>20</b>
Верно детектировать логотип получателя груза	Обязательно к выполнению	5
Доставить груз как можно ближе к центру логотипа получателя Груз должен остаться на крыше!	Обязательно к выполнению	$9 \cdot 2 = 18$
Посадить квадрокоптер на крышу с посадочной площадкой Квадрокоптер должен остаться на крыше!	Обязательно к выполнению	$7 \cdot 2 = 14$
Время выполнения заданий квадрокоптера		Лучшее время — 3 Второе время — 2 Третье время — 1
<b>Итого</b>		<b>40</b>

### *Оценка результатов и подведение итогов*

Максимальная сумма баллов за выполнение всех подзадач — 100 баллов.

В день финальных заездов участники получают баллы за выполнение элементов финального заезда. Максимальная сумма баллов за выполнение всех элементов финального заезда — 100 баллов.

Баллы, полученные за подзадачи, складываются с баллами, полученными за финальные заезды. Первое место занимает команда с наибольшей суммой баллов. Остальные команды располагаются за ней в порядке убывания суммы баллов.

Максимальная оценка, которую может получить команда — 200 баллов.

Итоговые результаты по профилю будут подведены 19 марта 2022 9:00–11:30 (МСК).

# Критерии определения победителей и призеров

## Первый отборочный этап

В первом отборочном этапе участники решали задачи по двум предметам: физика 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 70 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 более 75 баллов, чтобы пройти в финал.

## Заключительный этап

### *Индивидуальный предметный тур*

- Физика — максимально возможный балл за все задачи — 100 баллов;
- Информатика — максимально возможный балл за все задачи — 100 баллов.

### *Командный практический тур*

Команды, прошедшие в заключительный этап, получали за командный практический тур от 0 до 200 очков внутриигровых очков: команда, набравшая максимальное число очков, становилась командой-победителем.

В заключительном этапе олимпиады баллы участника складываются из двух частей, каждая из которых имеет собственный вес: баллы за индивидуальное решение задач по предметам (физика, информатика) с весом 0,15 каждый предмет и баллы за командное решение практических задач с весом 0,7.

Итоговый балл определяется по формуле:

$$S = 0,15 \cdot (S1 + S2) + 0,7 \cdot S3, \text{ где}$$

$S1$  — балл первой части заключительного этапа по физике в стобалльной системе ( $S1_{\max} = 100$ );

$S2$  — балл первой части заключительного этапа по информатике в стобалльной системе ( $S2_{\max} = 100$ );

$S3$  — итоговый балл командного тура ( $S3_{\max} = 100$ ).

---

Итого максимально возможный балл по условиям общего рейтинга:

$$0,15 \cdot (100 + 100) + 0,7 \cdot 100 = 100 \text{ баллов.}$$

***Критерий определения победителей и призеров (независимо от класса)***

Независимо от класса, был сделан общий рейтинг, где 9 классы участвовали на общих основаниях с 10–11 классами. С начала рейтинга были выбраны 2 победителя и 6 призеров (первые 25% участников рейтинга становятся победителями или призерами — первые 7% участников рейтинга становятся победителями, оставшиеся 18% — призерами).

Категория	Количество баллов
Победители	49 и выше
Призеры	от 16,5 до 49