Олимпиада школьников

ТИИМ - Технологии. Интеллект. Информатика. Математика

Задания заключительного тура с решениями.

2021-2022 учебный год

Информатика

▷ Задача 1. Активация портала

Космический корабль движется в направлении портала, отправляющего корабль на землю. Портал имеет форму тетраэдра. Автоматическая система должна активировать портал, как только корабль оказывается внутри(включая границу). Известны текущие координаты корабля и координаты вершин тетраэдра. Напишите программу, возвращающую “Активировать”, если корабль внутри портала, и “Еще не долетели”, если нет.

Входные данные:

Строки 1-4 – координаты вершин тетраэдра, разделенные пробелом, целые числа

Строка 5 – координаты корабля.

Выходные данные:

“Активировать”, если корабль внутри портала, и “Еще не долетели”, если нет

Ограничение времени выполнения программы: 1 секунда

Пример тестовых данных:

Входные данные	Результат работы программы
2 1 3 5 1 3 2 1 6 2 4 3 2 2 4	Активировать
-2 -7 -3 -2 -4 -3 -5 -4 -3 -2 -4 -6 -2 -5 -2	Еще не долетели
6 4 4 3 1 1 4 4 2 7 6 4 4 2 2 2	Активировать

Решение задачи на языке Python

Основная идея состоит в том, что если точка лежит внутри тетраэдра, то она находится по ту же сторону от плоскостей, в которых лежат грани пирамиды, что и противолежащие этим сторонам вершины.

Чтобы решить задачу, мы составляем уравнение плоскости по трем трем вершинам, подставляем в уравнение координаты точки, в которой находится корабль, и четвертой вершины. Если знаки для обеих точек одинаковы, значит они лежат по одну сторону от плоскости.

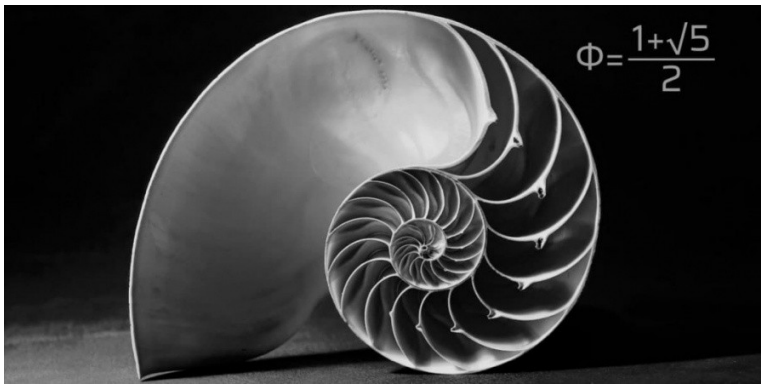
Если это условие выполняется для всех 4 граней, значит, точка находится внутри пирамиды.

```
# Подстановка точки в уравнение плоскости(вычисляем определитель)
```

```
def checkdot(plane,dot):
    res =\
    (dot[0] - plane[0][0])*(plane[1][1] - plane[0][1])*\
    (plane[2][2] - plane[0][2])+\
    (dot[1] - plane[0][1])*(plane[1][2] - plane[0][2])*\
    (plane[2][0] - plane[0][0])+\
    (dot[2] - plane[0][2])*(plane[1][0] - plane[0][0])*\
    (plane[2][1] - plane[0][1])-\
    (dot[2] - plane[0][2])*(plane[1][1] - plane[0][1])*\
    (plane[2][0] - plane[0][0])-\
    (dot[1] - plane[0][1])*(plane[1][0] - plane[0][0])*\
    (plane[2][2] - plane[0][2])-\
    (dot[0] - plane[0][0])*(plane[2][1] - plane[0][1])*\
    (plane[1][2] - plane[0][2])
    return res

c1 = list(map(int, input().split()))
c2 = list(map(int, input().split()))
c3 = list(map(int, input().split()))
c4 = list(map(int, input().split()))

d = list(map(int, input().split()))

result = 1

plane = [c4,c2,c3]
if checkdot(plane, d)*checkdot(plane,c1)<0:
    result = 0

plane = [c1,c2,c3]
if checkdot(plane, d)*checkdot(plane,c4)<0:
    result = 0

plane = [c1,c3,c4]
if checkdot(plane, d)*checkdot(plane,c2)<0:
    result = 0
```

```
plane = [c1,c2,c4]
if checkdot(plane, d)*checkdot(plane,c3)<0:
    result = 0

if result:
    print('Активировать')
else:
    print('Еще не долетели')
```

▷ Задача 2. Компьютер Фибоначчи



Фибоначчиева система счисления(ФСС) — смешанная система счисления для целых чисел на основе чисел Фибоначчи $F_2 = 1$, $F_3 = 2$, $F_4 = 3$, $F_5 = 5$, $F_6 = 8$ и т. д. Т.е. число в ФСС имеет вид последовательности нулей и единиц, аналогично двоичной системе счисления, но в качестве весов разряда вместо степеней двойки используются числа Фибоначчи. Эта запись не может иметь двух подряд идущих единиц.

Число	Запись в ФСС
0	$0 \ldots\ldots 0$
$F_2 = 1$	1
$F_3 = 2$	10
$F_4 = 3$	100
4	101
$F_5 = 5$	1000
6	1001
7	1010
$F_6 = 8$	10000

Любое целое положительное число может быть записано в ФСС, что следует из теоремы:

Теорема Цекендорфа: Любое неотрицательное целое число единственным образом представимо в виде суммы некоторого набора попарно различных чисел Фибоначчи с индексами, большими единицы, не содержащего пар соседних чисел Фибоначчи.

Сложение чисел в позиционных системах счисления выполняется с использованием переноса, позволяющего устранять последствия переполнения разряда. Например, в двоичной системе: $01 + 01 = 02 = 10$.

В фибоначчиевой системе счисления дело обстоит сложнее:

–Во-первых, вес старших разрядов не является кратным весу разряда, из которого требуется перенос.

–Во-вторых, требуется избавляться от соседних единиц

Для сложения чисел в ФСС будет полезным помнить, что

$F_k + F_k = F_k + F_{k-1} + F_{k-2}$, для $k > 3$

$F_k + F_k = F_{k+1} + F_{k-1}$, для $k = 3$

$F_k + F_k = F_{k+1}$, для $k = 2$

Задача

Вычислить сумму двух заданных чисел в ФСС, числа могут быть длиной до 200 знаков.

Для справки

В СССР занималась разработкой процессора на основе Фибоначчиевой системы счисления группа Стахова А.П.. Стахов Алексей Петрович - Украинский инженер и математик, внёсший вклад в развитие кибернетики и вычислительной техники. Разработал алгоритмическую теорию измерения, теорию р-кодов Фибоначчи и кодов золотой р-пропорции и основанные на них новые машинные арифметики, положенные в основу концепции компьютеров Фибоначчи, а также «математику гармонии» как новое междисциплинарное направление современной науки. Изобретения Стахова по направлению "Компьютеры Фибоначчи"защищены зарубежными патентами (65 патентов США, Японии, Англии, Франции, Германии, Канады и др. стран).

Входные данные:

Число 1

Число 2

Выходные данные:

Сумма чисел

Ограничение времени выполнения программы: 1 секунда

Пример тестовых данных:

Входные данные	Результат работы программы
1 1	10
1000 1000	10010
10100 100	100001

Решение задачи на языке Python

```
#Приведение числа к минимальной форме записи(без двух идущих подряд единиц)
```

```
def normalize(s):
    l = len(s)
    is_normal = 1

    for i in range(l-1):
        if s[i]==1 and s[i+1]==1:
            s[i-1] = 1
            s[i] = 0
            s[i+1] = 0
            if i>1 and s[i-2]==1:
                is_normal = 0
    if is_normal==0:
        normalize(s)
    return s

# Добавляем единицу( s - к какому числу добавляем,
# r- многоразрядный перенос, pos - позиция, в которую добавляем)
def fib_add(s,r,pos):
    if s[pos]==0:
        s[pos]=1
    else:
        r[pos]=1

#функция для вычисления суммы (a, b - числа в ФСС, l - длина)
def fib_sum(a,b,l):
    s = [0]*(l+1)
    r = [0]*(l+1)
    for i in range(l,-1,-1):
        s[i]=s[i]+a[i]+b[i]
        if s[i]==2:
            if i==l:
                s[i-1]=1
                s[i]=0
            if i==l-1:
                fib_add(s,r,i+1)
                s[i-1]=1
```

```
                s[i]=0
            if i<l-1:
                fib_add(s,r,i+1)
                fib_add(s,r,i+2)
                s[i]=1
    normalize(s)
    if r.count(1)>0:
        s = fib_sum(s[:],r[:],l)
    return s

#получаем входные данные
a = list(map(int,list(input())))
b = list(map(int,list(input())))

#максимально возможная длина суммы. В ФСС может добавиться два разряда.
l = max(len(a),len(b))+2

a = [0]*(l - len(a))+[0]+a
b = [0]*(l - len(b))+[0]+b

s = fib_sum(a[:],b[:],l)

while s[0] == 0:
    s = s[1:]

print(''.join(map(str,s)))
```

▷ Задача 3. Интеллектуальное такси

Ни для кого не секрет, что сейчас о каждом человеке в разных базах данных, на сайтах и сервисах, хранится очень большое количество информации: данные перемещений, покупки, любимые сайты, просмотренная реклама, переходы по ссылкам, лайки, фотографии, поисковые запросы.

В сверхбыстрое такси сели 4 человека. Искусственный интеллект должен определить, куда их везти, основываясь на информации, которую он нашел в своей базе данных. На вход программы передаются 7 фраз с утверждениями о людях, городах и цветах, на выходе программа выводит, в какой город и к дому какого цвета отвезти пассажира. Ответ искусственного интеллекта должен быть отсортирован по имени по алфавиту.

Интересный факт

В 2015 году было проведено исследование, которое показало, что анализ специальной программой всего 10 “лайков” позволяет программе узнать человека лучше, чем его знают его сослуживцы. По 70 “лайкам” программа узнает о человеке столько же, как его близкий друг или сосед по комнате. Уровню 150 “лайков” соответствует уровень знания человека родителями, братьями или сестрами, а анализ 300 и более лайков позволяет программе узнать человека лучше, чем его знает супруга или супруг.

Входные данные:

7 строк с найденными данными(см пример)

Выходные данные:

4 строки с ответами в формате

“Нужно отвезти пассажира *Имя* в г. *Название города*, *цвет дома* дом”

Возможные варианты цветов, имен, городов

Цвета - красный, зеленый, оранжевый, желтый, белый, черный, голубой, фиолетовый, синий, пурпурный

Имена - Саша, Алиса, Маша, Петр, Иван, Леша, Женя, Юля, Оля, Катя, Даша, Дима, Наташа, Сережа, Вера, Денис

Города - Москва, Самара, Омск, Белгород, Нижний Новгород, Воронеж, Томск, Кемерово, Казань, Улан-Удэ, Шахтерск, Санкт-Петербург,

Ульяновск, Красноярск

Ограничение времени выполнения программы: 1 секунда

Примеры тестовых данных:

Входные данные	Результат работы программы
Алиса никогда не была в г. Самара В г. Омск не разрешено красить дома в красный Женя покрасил дом в синий цвет Маша не любит зеленый цвет Петр часто звонит другу из г. Москва, у которого дом желтый Женя живет в г. Белгород Петр больше не живет в г. Омск	Нужно отвезти пассажира Алиса в г. Омск, зеленый дом Нужно отвезти пассажира Женя в г. Белгород, синий дом Нужно отвезти пассажира Маша в г. Москва, желтый дом Нужно отвезти пассажира Петр в г. Самара, красный дом
Саша часто звонит другу из г. Москва, у которого дом желтый Юля никогда не была в г. Самара Даша не любит зеленый цвет Иван живет в г. Белгород В г. Омск не разрешено красить дома в оранжевый Саша больше не живет в г. Омск Иван покрасил дом в синий цвет	Нужно отвезти пассажира Даша в г. Москва, желтый дом Нужно отвезти пассажира Иван в г. Белгород, синий дом Нужно отвезти пассажира Саша в г. Самара, оранжевый дом Нужно отвезти пассажира Юля в г. Омск, зеленый дом

Решение задачи на языке Python

Чтобы установить соответствие людей, городов и цветов, нам понадобятся три таблицы соответствия. Таблицы будут изначально заполнены нулями, в процессе анализа входных данных будем ставить 1 в таблицу в случае соответствия и -1, если соответствия нет.

При этом в процессе заполнения таблиц проверяем, если в клетке стоит 1, то в строке и столбце все остальные значения будут равны -1. Ес-

ли в трех клетках строки/столбца стоит -1, значит в четвертой - единица. Также не забываем про связь между таблицами: например, если известно соответствие город-человек и город-цвет, значит, мы можем заполнить клетку в таблице человек-цвет.

```
colors =['красный', 'зеленый', 'оранжевый', 'желтый', 'белый', 'черный',\
 'голубой', 'фиолетовый', 'синий', 'пурпурный']

names = ['Саша', 'Алиса', 'Маша', 'Петр', 'Иван', 'Леша', 'Женя', 'Юля',\
 'Оля', 'Катя', 'Даша', 'Дима', 'Наташа', 'Сережа', 'Вера', 'Денис']

cities = ['г. Москва', 'г. Самара', 'г. Омск', 'г. Белгород',\
 'г. Нижний Новгород', 'г. Воронеж', 'г. Томск', 'г. Кемерово',\
 'г. Казань','г. Улан-Удэ','г. Шахтерск', 'г. Санкт-Петербург',\
  'г. Ульяновск','г. Красноярск']

names_cities = [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]
cities_colors =[[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]
names_colors = [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]

#Функция для выделения используемых значений из условий
def getUsed(statements, values):
    used = [];
    for s in statements:
        for v in values:
            if s.find(v)>=0 and used.count(v)==0:
                used.append(v)
    return used

# Устанавливает значение val в заданной клетке заданной таблицы связности
def setValue(arr,i,j,val):
    if (val == 1):
        for c in range(4):
            arr[i][c]=-1
            arr[c][j]=-1
    arr[i][j] = val
    check_row(arr,i)
```

```
    check_col(arr,j)

#Дозаполняем строчку, если в ней три раза -1
def check_row(arr,i):
    if sum(arr[i])==-3:
        setValue(arr,i,arr[i].index(0),1)

#Дозаполняем столбец, если в нем три раза -1
def check_col(arr,i):
    s =0
    for j in range(4):
        s+=arr[j][i]
    if s==-3:
        for j in range(4):
            if arr[j][i]==0:
                setValue(arr,j,i,1)



statements = []

for i in range(7):
    s = input()
    statements.append(s)

#Первым делом выделяем из условий имена, цвета и города, которые там упоминаются
usedColors = getUsed(statements,colors)
usedCities = getUsed(statements,cities)
usedNames = getUsed(statements,names)

#Сразу сортируем имена по алфавиту, для корректного вывода результата
usedNames.sort()

#Заполняем таблицу сведениями из высказываний
for s in statements:
    nameInd=-1
```

```
    colorInd=-1
    cityInd=-1
    for i in range(4):
        if s.find(usedNames[i])>=0:
            nameInd = i
        if s.find(usedColors[i])>=0:
            colorInd = i
        if s.find(usedCities[i])>=0:
            cityInd = i

# Если в одном высказывании сразу и город и имя и цвет
    if (nameInd>=0 and colorInd>=0 and cityInd>=0):
        setValue(names_colors,nameInd,colorInd,-1)
        setValue(cities_colors,cityInd,colorInd,1)
        setValue(names_cities,nameInd,cityInd,-1)
    # Если только два параметра
    else:
        if(nameInd>=0) and (cityInd>=0):
            if s.find(' не ')>=0:
                setValue(names_cities,nameInd,cityInd,-1)
            else:
                setValue(names_cities,nameInd,cityInd,1)

        if((colorInd>=0) and (cityInd>=0)):
            if s.find(' не ')>=0:
                setValue(cities_colors,cityInd,colorInd,-1)
            else:
                setValue(cities_colors,cityInd,colorInd,1)

        if(nameInd>=0 and colorInd>=0):
            if s.find(' не ')>=0:
                setValue(names_colors,nameInd,colorInd,-1)
            else:
                setValue(names_colors,nameInd,colorInd,1)

#Дозаполняем таблицы, учитывая связи между ними
```

```
for i in range(4):
    for j in range(4):
        for k in range(4):
            v = names_cities[i][k]*names_colors[i][j]
            if v!=0 and names_cities[i][k]+names_colors[i][j]>-2:
                setValue(cities_colors,k,j, v)

            v =  names_cities[k][i]*cities_colors[i][j]
            if v!=0 and names_cities[k][i]+cities_colors[i][j]>-2:
                setValue(names_colors,k,j,v)

            v = cities_colors[k][j]*names_colors[i][j]
            if v!=0 and cities_colors[k][j]+names_colors[i][j]>-2:
                setValue(names_cities,i,k,v)


#Формируем ответ, выбирая города и цвета для имен из массива usedNames
for i in range(4):
    cityInd = names_cities[i].index(1)
    colorInd  = names_colors[i].index(1)

    print('Нужно отвезти пассажира '+usedNames[i]+' в '+usedCities[cityInd]+\
    '\, '+usedColors[colorInd]+' дом')
```

▷ Задача 4. Реверс-инжиниринг

Инженерам попала в руки программа - черный ящик. На вход подаются два числа и известен итоговый результат. Неизвестно, как она этого добивается, но поставлена задача создать программу, выдающую тот же результат на тех же числах.

Входные данные:

a, b - целые неотрицательные числа

Выходные данные:

результат

Ограничение времени выполнения программы: 1 секунда

Пример тестовых данных:

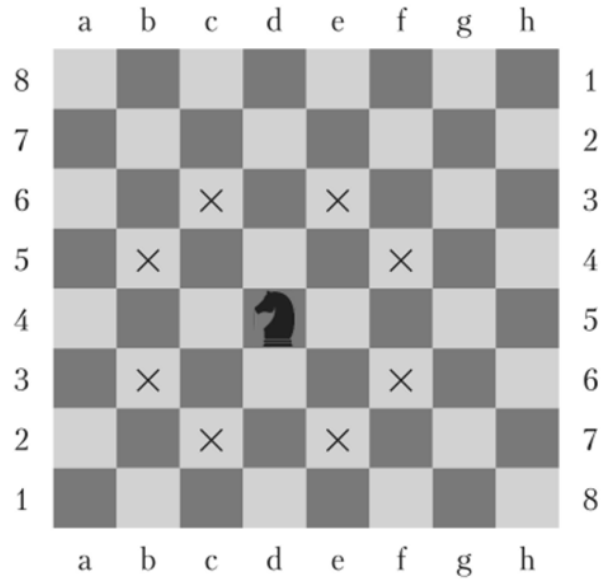
Входные данные	Результат работы программы
1 0	1
2 1	3
54 81	119

Решение задачи на языке Python

Участникам предлагалось 10 тестовых примеров с ответами, по которым можно было воссоздать функцию, выдающую требуемый результат.

```
(a,b)=list(map(int,list(input().split(' '))))
print(a|b)
```

▷ Задача 5. Ход конем



Саша и Женя учатся играть в шахматы и хотят использовать для них свои навыки программирования. Руководитель кружка предложил им решить следующую задачу:

Задание:

Найти минимальное количество ходов, которое понадобится шахматному коню, чтобы добраться с одной клетки до другой

Входные данные:

Клетка1 клетка2

Выходные данные:

число - минимальное количество ходов

Ограничение времени выполнения программы: 1 секунда

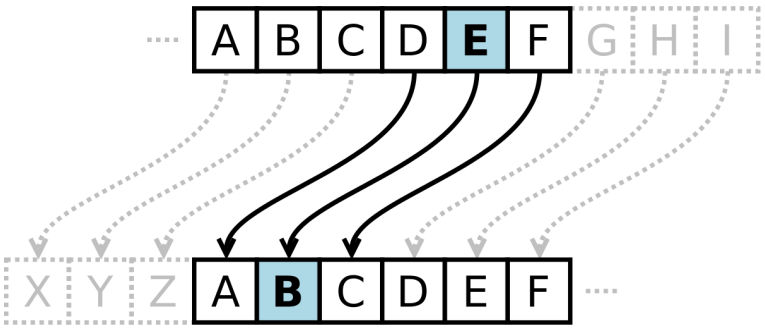
Пример тестовых данных:

Входные данные	Результат работы программы
a1 b1	3
a2 f4	3
a2 f5	4

Решение задачи на языке Python

```
from collections import deque
moves = ((1, 2), (1, -2), (-1, 2), (-1, -2), (2, 1), (2, -1), (-2, 1), (-2, -1))

def knight(p1, p2):
    x, y = ord(p2[0])-97, int(p2[1])-1
    left, seen = deque([(ord(p1[0])-97, int(p1[1])-1, 0)]), set()
    while left:
        i, j, v = left.popleft()
        if i==x and j==y: return v
        if (i, j) in seen: continue
        seen.add((i, j))
        for a,b in moves:
            if 0 <= i+a < 8 and 0 <= j+b < 8:
                left.append((i+a, j+b, v+1))

(p1,p2) = input().split(' ')

print(knight(p1,p2))
```

▷ Задача 6. Письма Цезаря



Археологи нашли новые, ранее неизвестные письма Цезаря. Известно, что Цезарь отправлял письма в зашифрованном виде и всегда подписывался своим именем. Письма написаны на английском. Сдвиг шифра может быть разным в каждом сообщении. Знаки препинания, пробелы и специальные символы остаются без изменений, регистр символов сохраняется. Если в сообщении нет подписи Цезаря - вывести 'No Julius'

Шифр Цезаря со сдвигом на 3:

A заменяется на **D**,

B заменяется на **E**,...

и так далее...

Z заменяется на **C**.

Часто для удобства использования шифра Цезаря используют два насаженных на общую ось диска разного диаметра с нарисованными по краям дисков алфавитами. Изначально диски поворачиваются так, чтобы напротив каждой буквы алфавита внешнего диска находилась та же буква алфавита малого диска. Если теперь повернуть внутренний диск на несколько символов, то мы получим соответствие между символами внешнего диска и внутреннего — шифр Цезаря. Получившийся диск можно использовать как для шифрования, так и для расшифровки.

Например, если внутреннее колесо повернуть так, чтобы символу A внешнего диска соответствовал символ D внутреннего диска, то мы получим шифр со сдвигом 3 влево.

Забавное замечание

Неизвестно, насколько эффективным шифр Цезаря был в то время,

но, вероятно, он был разумно безопасен, не в последнюю очередь благодаря тому, что большинство врагов Цезаря было неграмотным, и многие предполагали, что сообщения были написаны на неизвестном иностранном языке

Входные данные:

Зашифрованная строка

Выходные данные:

Расшифрованная строка

Ограничение времени выполнения программы: 1 секунда

Пример тестовых данных:

Входные данные	Результат работы программы
Rzzo wfnv! Ufwtfd	Good luck! Julius
Znoy oy tuz znk jxuojy eua'xk ruuqotm lux. Paroay	This is not the droids you're looking for. Julius
I am full of surprises, don't I? Julius	I am full of surprises, don't I? Julius
What's going on?	No Julius

Решение задачи на языке Python

```
def encrypt(text,s):
    result = ""

    # traverse text
    for i in range(len(text)):
        char = text[i]

        # Encrypt uppercase characters
        if (char.isupper()):
            result += chr((ord(char) + s-65) % 26 + 65)
```

```
        # Encrypt lowercase characters
        else:
            if (char.islower()):
                result += chr((ord(char) + s - 97) % 26 + 97)
            else:
                result+=char

    return result

def decipher(s,k):
    i=0
    while s.find('Julius')!=l-6:
        if i==26:
            return 'No Julius'
        s = encrypt(s,1)
        i +=1
    else:
        return s

s = input()
l = len(s)
signPos = l - 6

print(decipher(s,signPos))
```