Технологии дополненной реальности

2022/23 учебный год

Заключительный этап

Предметный тур

Информатика. 8–11 класс

Задача VI.1.1.1. Переписка в чате (10 баллов)

Марик и Ярик обсуждают идею проекта. Для истории они решили распечатать свою переписку, но сомневаются, что хватит бумаги. Напишите программу, чтобы определить, сколько строк займёт переписка, если известно число символов в строке W и слова можно переносить только целиком и нет слов длиннее, чем W. Каждая реплика печатается с новой строки. Перенос строки аналогичен пробелу, поэтому сохранять пробел между соседними словами на разных строках не нужно. Значения W и N не превышают 10000.

Формат входных данных

Число N строк текста (реплик в переписке), ширина W страницы в символах.

В следующих строках N реплик.

Формат выходных данных

Целое число — количество строк.

Примеры

Пример №1

Стандартный ввод
4 8 How are you? Fine, thanks! Let's return to the work. Agree!

Стандартный вывод
9

Решение

Получаем на вход N и W.

Далее читаем каждую строку и разбиваем её на слова. Смотрим каждое слово. Если длина строки (переменная s) равна нулю, то длина текущей строки будет равно длине слова. Если же сумма длины текущей строки и нового слова меньше W — слово влазит в текущую строку, мы просто увеличиваем длину строки. Иначе же увеличиваем переменную `lines` (количество строк) и s приравниваем к длине текущего слова (т. е. мы сделали перенос). В конце перебора слов очередной строки смотрим, если длина текущей строки больше 0 — нужно сделать перенос, чтобы следующая реплика была с новой строки.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, w = map(int, input().split())

lines = 1
s = 0

for i in range(n):
    words = input().split()
    for word in words:
        if s == 0:
            s = len(word)
            continue
        if s + len(word) + 1 <= w:
            s += len(word) + 1
        else:
            lines += 1
            s = len(word)

    if s != 0:
        lines += 1
        s = 0
print(lines - 1)
```

Задача VI.1.1.2. Склад Деда Мороза (проверка) (10 баллов)

Дед Мороз пытается оптимизировать доставку подарков на своём складе прямоугольной формы $W \cdot H$ ($W, H < 20$). Число подарков равно числу мест на складе ($N = W \cdot H$). Вход на склад только один и расположен в верхнем левом углу (то есть прилегает к координате 1, 1). Каждый подарок имеет рейтинг срочности от 1 до N (рейтинг 1 соответствует наибольшей срочности). Самый срочный подарок располагается на месте с координатами (1, 1). Важно расположить их так, чтобы подарок с большей срочностью находился не дальше от входа, чем подарок с меньшей срочностью. Расстоянием между двумя местами (W_1, H_1) и (W_2, H_2) считается значение $|W_1 - W_2| + |H_1 - H_2|$. Вам дано расположение подарков на складе в виде прямоугольной матрицы. Проверьте, соответствует ли расстановка срочности каждого подарка.

Формат входных данных

В первой строке даны целые числа W и H — размеры склада. Далее следуют H строк со значением рейтинга каждого подарка.

Формат выходных данных

Выведите 1, если подарки расставлены в соответствии с рейтингом и 0 в противном случае.

Примеры

Пример №1

Стандартный ввод
2 2 1 2 3 4
Стандартный вывод
1

Решение

Получаем на вход W и H. Далее в двумерный список `hall` заносим расстановку подарков в комнате.

Затем генерируем словарь `amount`, в ключах которого хранится расстояние от входа, а в значениях — сколько раз оно встречается в нашей комнате. Затем сортируем `amount` по ключам и получаем `sorted_amounts`.

В словарь `to_use` заносим номера срочности подарков, которые могут находится на расстоянии `num`.

Далее проверяем, соответствует ли расстояние модарка номеру его срочности.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
w, h = map(int, input().split())

hall = []

for i in range(h):
    hall.append(list(map(int, input().split())))

amounts = {}

for i in range(h):
    for j in range(w):
        if i + j not in amounts:
            amounts[i + j] = 0
        amounts[i + j] += 1

myKeys = list(amounts.keys())
myKeys.sort()
sorted_amounts = {i: amounts[i] for i in myKeys}

to_use = {}
```

```
21
22  l = 1
23  for num in sorted_amounts:
24      to_use[num] = []
25      while sorted_amounts[num] > 0:
26          to_use[num].append(l)
27          l += 1
28          sorted_amounts[num] -= 1
29  OK = True
30  for i in range(h):
31      for j in range(w):
32          if hall[i][j] in to_use[i + j]:
33              to_use[i + j].remove(hall[i][j])
34          else:
35              OK = False
36
37  if OK:
38      print(1)
39  else:
40      print(0)
```

Задача VI.1.1.3. Склад Деда Мороза (расстановка) (20 баллов)

Дед Мороз пытается оптимизировать доставку подарков на своём складе прямоугольной формы $W \cdot H$ ($W, H < 20$). Число подарков равно числу мест на складе ($N = W \cdot H$). Вход на склад только один и расположен в верхнем левом углу (то есть прилегает к координате 1, 1). Каждый подарок имеет рейтинг срочности от 1 до N (рейтинг 1 соответствует наибольшей срочности). Самый срочный подарок располагается на месте с координатами (1, 1). Теперь необходимо разместить все подарки. Важно расположить их так, чтобы подарок с большей срочностью находился не дальше от входа, чем подарок с меньшей срочностью. Расстоянием между двумя местами (W_1, H_1) и (W_2, H_2) считается значение $|W_1 - W_2| + |H_1 - H_2|$.

Формат входных данных

Каждый из тестов описывается целыми числами W и H, разделённых пробелами.

Формат выходных данных

Выведите, как будет выглядеть склад после размещения подарков.

Выведите H строк, в каждой по W чисел, j-е число i-й строки должно быть рейтингом подарка, расположенного на месте (i, j).

Если подходящих способов несколько, выведите любой из них. В левом верхнем углу подарок с рейтингом 1.

Примеры

Пример №1

Стандартный ввод
2 3

Стандартный вывод
1 3 2 4 5 6

Решение

Читаем W и H. Заполняем массив `hall` (зал с подарками) нулями.

Затем генерируем словарь `amount`, в ключах которого хранится расстояние от входа, а в значениях — сколько раз оно встречается в нашей комнате.

Затем сортируем amount по ключам и получаем `sorted_amounts`.

В словарь `to_use` заносим номера срочности подарков, которые могут находится на расстоянии `num`.

Теперь берём из `to_use` числа и заносим их в `hall`, в соответсвии с расстоянием. Выводим `hall` на экран.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
w, h = map(int, input().split())

hall = [[0 for i in range(w)] for j in range(h)]

amounts = {}

for i in range(h):
    for j in range(w):
        if i + j not in amounts:
            amounts[i + j] = 0
        amounts[i + j] += 1

myKeys = list(amounts.keys())
myKeys.sort()
sorted_amounts = {i: amounts[i] for i in myKeys}

to_use = {}

l = 1
for num in sorted_amounts:
    to_use[num] = []
    while sorted_amounts[num] > 0:
        to_use[num].append(l)
        l += 1
        sorted_amounts[num] -= 1

```

```
for i in range(h):
    for j in range(w):
        hall[i][j] = to_use[i + j][-1]
        to_use[i + j].pop(-1)

for i in range(h):
    for j in range(w):
        print(hall[i][j], end=" ")
    print()
```

Задача VI.1.1.4. Контур для юнитов (30 баллов)

В стратегических играх часто требуется выделить несколько единиц техники (`units`, юнитов) для групповых действий. Вам известны координаты X, Y каждого юнита. Требуется построить контур (ломаную линию) через минимальное число юнитов так, чтобы все остальные оказались внутри полученного многоугольника.

Формат входных данных

Число юнитов, далее в отдельных строках координаты X, Y каждого из них, разделённых пробелом (целые числа).

Юниты пронумерованы с 0 в порядке перечисления во входных данных. Гарантируется, что юниты не лежат на одной прямой.

Формат выходных данных

Номера юнитов, которые образуют контур. Набор целых чисел, разделённых пробелом (упорядоченные по возрастанию).

Примеры

Пример №1

Стандартный ввод
5 0 0 5 0 0 5 1 1 2 2

Стандартный вывод
0 1 2

Решение

Для решения этой задачи можно использовать алгоритм `Gift wrapping` (или `Jarvis March`), который ищет выпуклую оболочку множества точек. Идея алгоритма заключается в следующем:

1. Находим самую левую точку (самую нижнюю в случае, если есть несколько самых левых).
2. Добавляем эту точку в контур.
3. Находим следующую точку контура: она должна иметь наименьший полярный угол относительно текущей точки (то есть наименьший угол между вектором, идущим из текущей точки в новую точку, и осью Ox).
4. Добавляем найденную точку в контур и повторяем шаги 3–4, пока не вернёмся в начальную точку.

В результате получится выпуклый многоугольник, образованный точками контура.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
ans = []

class Point:
    def __init__(self, x, y, n):
        self.x = x
        self.y = y
        self.n = n

def Left_index(points):
    '''
    Finding the left most point
    '''
    minn = 0
    for i in range(1, len(points)):
        if points[i].x < points[minn].x:
            minn = i
        elif points[i].x == points[minn].x:
            if points[i].y > points[minn].y:
                minn = i
    return minn

def orientation(p, q, r):
    '''
    To find orientation of ordered triplet (p, q, r).
    The function returns following values
    0 --> p, q and r are collinear
    1 --> Clockwise
    2 --> Counterclockwise
    '''
    val = (q.y - p.y) * (r.x - q.x) - \
          (q.x - p.x) * (r.y - q.y)

    if val == 0:
        return 0
    elif val > 0:
        return 1
    else:
        return 2

def convexHull(points, n):
    # There must be at least 3 points
```

```
    if n < 3:
        return

    # Find the leftmost point
    l = Left_index(points)

    hull = []

    '''
    Start from leftmost point, keep moving counterclockwise
    until reach the start point again. This loop runs O(h)
    times where h is number of points in result or output.
    '''
    p = l
    q = 0
    while (True):

        # Add current point to result
        hull.append(p)

        '''
        Search for a point 'q' such that orientation(p, q,
        x) is counterclockwise for all points 'x'. The idea
        is to keep track of last visited most counterclock-
        wise point in q. If any point 'i' is more counterclock-
        wise than q, then update q.
        '''
        q = (p + 1) % n

        for i in range(n):

            # If i is more counterclockwise
            # than current q, then update q
            if (orientation(points[p],
                            points[i], points[q]) == 2):
                q = i

        '''
        Now q is the most counterclockwise with respect to p
        Set p as q for next iteration, so that q is added to
        result 'hull'
        '''
        p = q

        # While we don't come to first point
        if (p == l):
            break

    # Print Result
    for each in hull:
        ans.append(points[each].n)

points = []

# Driver Code
n = int(input())
for i in range(n):
    data = input().split()
    points.append(Point(int(data[0]), int(data[1]), i))

```

```
convexHull(points, len(points))

ans.sort()

for a in ans:
    print(a, end=" ")
```

Задача VI.1.1.5. Колонии бактерий (30 баллов)

Молодой учёный Пётр выращивает культуры разных полезных бактерий. Так как подсчитать число бактерий в образце невозможно, он ориентируется на оптическую плотность, которую измеряет прибором каждый день. Чем выше плотность, тем больше бактерий. Пётр смог установить закон, по которому растут разные культуры. Предполагая, что на i-ый день оптическая плотность составляет ni, на следующий день её можно узнать по формуле $n_i + 1 = a \cdot n_i - b \cdot n_i^2$. Пётр интересуется, каково будущее каждой колонии бактерий, если их выращивать неограниченно долго?

Формат входных данных

На вход программа получает три вещественных числа: начальную оптическую плотность, коэффициенты a и b. Число знаков после запятой не более трёх. Диапазон входных данных: $0 \leqslant a, b, n < 10$.

Формат выходных данных

Сообщите предел, к которому стремится оптическая плотность колонии с округлением до 3 знаков после запятой. Если предела нет (плотность растёт неограниченно), выведите `-1.000`.

Примеры

Пример №1

Стандартный ввод
`0.500 1.500 1.000`
Стандартный вывод
`0.500`

Решение

Для решения задачи можно использовать итерационный процесс. Начиная с начальной оптической плотности, вычисляем плотность на следующий день, затем на следующий после него и так далее, до тех пор, пока не будет достигнут предел. Предел может быть достигнут в том случае, если последовательность сходится к какому-то конечному значению или ограничена сверху. В противном случае, когда последовательность не имеет предела, выводим `-1.000`.

Код работает следующим образом:

- Считываем три вещественных числа n, a, b с помощью функции `map`.

- Используем переменную `prev` для хранения текущей оптической плотности итерационного процесса.
- В цикле `for` вычисляем оптическую плотность на следующий день с помощью формулы из условия.
- Если разница между текущей и предыдущей оптической плотностью меньше определенной точности (`1e-9`), то мы достигли предела, и выводим значение оптической плотности, округленное до трех знаков после запятой, с помощью функции `format` и строкового метода `format`.
- Если количество итераций достигло максимума (999999), то мы не достигли предела, и выводим `-1.000`.
- Обновляем переменную `prev` на значение `next` и продолжаем итерационный процесс.

Обратите внимание, что мы ограничиваем количество итераций цикла `for` значением 1000000. Это необходимо для того, чтобы избежать бесконечного цикла в случае, если последовательность не имеет предела. Если программа достигает этого лимита итераций, то мы считаем, что последовательность не имеет предела.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, a, b = map(float, input().split())

prev = n
for i in range(1000000): # ограничиваем количество итераций
    next = a * prev - b * prev ** 2
    if abs(next - prev) < 1e-9: # достигнут предел
        print("{:.3f}".format(next))
        break
    elif i == 999999: # не достигнут предел
        print("-1.000")
    prev = next
```

Тестовые наборы для задач представленны по ссылке — https://disk.yandex.ru/d/SC7-OX2e3faC2A.

Математика. 8–9 классы

Задача VI.1.2.1. (20 баллов)

В игре Pokemon Go вероятность встретить покемона за 20-минутный интервал времени равна 0,9375. Найти вероятность его появления в следующие 5 минут.

Решение

Вероятность того, что за четыре интервала по 5 минут ни один покемон не появится, равна ;. И она же равна $(1-x)^4$, если обозначить за x вероятность того, что за 5 минут появится один покемон. Приравнивая эти величины, находим, что $x = 1 - \sqrt[4]{0,0635} = 0,5$. Это и есть искомая вероятность.

Ответ: 0,5.

Критерии оценивания

- Приведен правильный ответ, но не верно обоснован — 5 баллов.
- Приведен правильный ход решения, но получен не верный ответ — 10 баллов.
- Приведен правильный ответ, который полностью математически обоснован — 15 баллов.

Задача VI.1.2.2. (20 баллов)

Игровая зона имеет острый угол между стенами, которые отделяют ее от дорог. Внутри игровой зоны посажены два дерева: D_1 и D_2. Дети придумали игру, в которой они стартуют от первого дерева, добегают до ближайшей к нему стены l_1, касаются ее, затем добегают до второго дерева, касаются его, оттуда до стены l_2, и обратно до D_1. Кто первым вернется к D_1, тот и победил. Понятно, что стену можно касаться в любой ее точке. До каких точек на стенах l_1 и l_2 следует добегать, чтобы полный путь быть минимален, и игрок выиграл, т. е. первым вернулся к D_1?

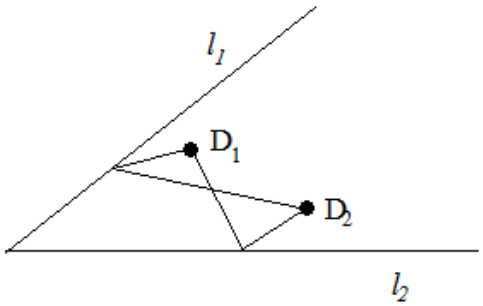

Рис. VI.1.1. Чертеж к задаче VI.1.2.2

Решение

Проложим перпендикуляр от D_1 к l_1 и продолжим его на такое же расстояние за стену до точки D_1'. Точку пересечения перпендикуляра и стены обозначим за N_1. Кратчайшее расстояние от D_1' до D_2 — по прямой D_2D_1'. Обозначим точку пересечения этой прямой со стеной за M_1.

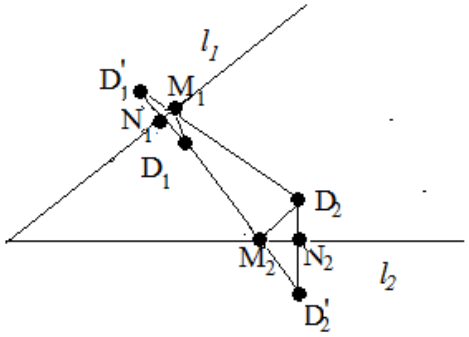

Рис. VI.1.2. Обозначения и дополнительные построения к задаче VI.1.2.2

Аналогично, проложим перпендикуляр от D_2 к l_2 и продолжим его на такое же расстояние за стену до точки D_2'. Точку пересечения перпендикуляра и стены обозначим за N_2. Кратчайшее расстояние от D_2' до D_1 – по прямой D_1D_2'. Обозначим точку пересечения этой прямой со стеной за M_2.

По построению, прямоугольные треугольники $D_1N_1M_1$ и $D_1'N_1M_1$ равны по двум сторонам и углу между ними. Следовательно, $D_1M_1 = D_1'M_1$, и когда человек пробегает расстояние от D_1 до M_1 плюс расстояние от M_1 до D_2, он пробегает такое же расстояние, что от D_1' до D_2, и это кратчайшее расстояние от первого дерева до второго с пробегом до стены. Нужно лишь «пересадить» дерево за стену и представить, что стены не существует. Аналогично со вторым деревом. Получили, что суммарный путь по ломанной $D_1M_1D_2M_2D_1$ эквивалентен пути $D_1'D_2$ плюс $D_2'D_1$, т. е. двум кратчайшим расстояниям: от «отражения» первого дерева за стену до второго и от «отражения» второго дерева за стену до первого.

Ответ: нужно касаться стен в точках, где стену пересекает воображаемая прямая, проложенная от точки, находящейся за стеной и симметричной положению ближайшего к стене дерева относительно стены, до другого дерева.

Критерии оценивания

- Приведен правильный ответ, но не обоснован — 10 баллов.
- Приведен правильный ответ, который полностью математически обоснован — 20 баллов.

Задача VI.1.2.3. (20 баллов)

В компьютерной игре нужно построить фонтан, такой, чтобы под ним могли пробегать персонажи игры, не попав под струю. Стойка фонтана должна быть направлена вертикально вверх. С вершины своей стойки фонтан должен бить по параболе так, чтобы наивысшая точка струи располагалась на высоте 3 см от уровня земли и

на расстоянии 1 см от стойки фонтана. Какой высоты должна быть стойка фонтана, чтобы струя достигала уровня земли на расстоянии не менее трех, и не более четырех см от стойки фонтана?

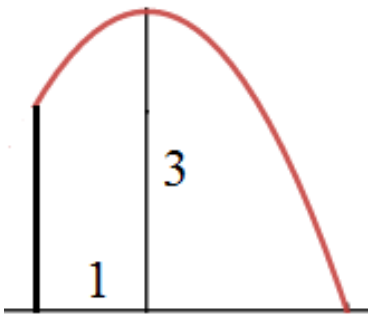

Рис. VI.1.3. Обозначения и Чертеж к задачее VI.1.2.3

Решение

Обозначим за ось x ось поверхности земли, и впишем фонтан в Декартову систему координат на плоскости так, как показано на рис.VI.1.4.

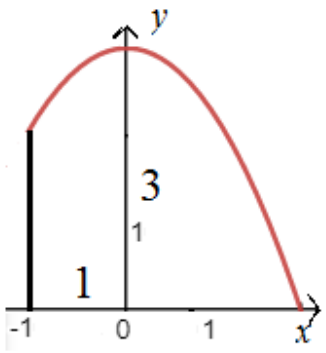

Рис. VI.1.4. Обозначения и дополнительные построения к задаче VI.1.2.3

Тогда уравнение параболы, описывающей поведение струи, будет следующее: $y = -ax^2 + 3$, где $a > 0$ — параметр, показывающий степень пологости («ширины») параболы. Струя достигает поверхности земли в точке, где $y = 0$. Решим уравнение $0 = -ax^2 + 3$, его положительное решение равно $x = \sqrt{\frac{3}{a}}$. Нас интересует, когда оно будет от 2 до 3 см: $2 \leqslant \sqrt{\frac{3}{a}} \leqslant 3$. Возведя неравенство в квадрат, имеем: $4 \leqslant \frac{3}{a} \leqslant 9$, перевернув дроби получим неравенство: $\frac{1}{9} \leqslant \frac{a}{3} \leqslant \frac{1}{4}$. Откуда окончательно получим, что $\frac{1}{3} \leqslant a \leqslant \frac{3}{4}$.

Для двух крайних случаев найдем на параболе точку, соответствующую $x = -1$, это и будет высота стойки фонтана. При $a = \frac{1}{3}$ уравнение параболы будет

$$y = -\frac{1}{3}x^2 + 3, \text{ а } y(-1) = 3 - \frac{1}{3} = 2\frac{2}{3}.$$

При $a = \frac{3}{4}$ уравнение параболы будет

$$y = -\frac{3}{4}x^2 + 3, \text{ а } y(-1) = 3 - \frac{3}{4} = 2\frac{1}{4}.$$

Следовательно, для того, чтобы струя достигала уровня земли на расстоянии не менее трех, и не более четырех см от стойки фонтана, нужно чтобы стойка фонтана была от $2\frac{1}{4}$ до $2\frac{2}{3}$ см высотой.

Ответ: от $2\dfrac{1}{4}$ до $2\dfrac{2}{3}$ см.

Критерии оценивания

- Верно составлено и верно решено уравнение для нахождения всех подходящих парабол, но способ нахождения высоты стойки не указан — 5 балла.
- Верно составлено уравнение для нахождения всех подходящих парабол. Верно показано, как находить высоту стойки, но какой-нибудь из этих параметров найден неверно из-за арифметической ошибки — 10 баллов.
- Задача решена верно и полностью математически обоснована — 20 баллов.

Задача VI.1.2.4. (40 баллов)

В игре Minecraft у прямоугольного заграждения 10×20 привязана коза. Она привязана к одному из углов заграждения снаружи прямоугольника на веревку длины l. Внутрь заграждения она попасть не может, и может пастись только снаружи, насколько позволит веревка. Какова площадь выпаса для козы, если:

а) длина веревки $l = 5$,

б) длина веревки $l = 25$,

в) длина веревки $l = 50$?

Решение

а) Площадь выпаса равна 3/4 круга радиуса 5 (см. рис. VI.1.5). Следовательно, площадь выпаса равна $S = \dfrac{3}{4}\pi R^2 = \dfrac{75\pi}{4} \approx 58,90486$.

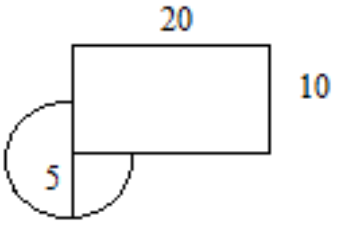

Рис. VI.1.5. Чертеж к задаче VI.1.2.4а)

б) Площадь выпаса равна 3/4 круга радиуса 25, 1/4 круга радиуса 15 и 1/4 круга радиуса 5 (см. рис. VI.1.6). Следовательно, площадь выпаса равна $S = \dfrac{3}{4}\pi(25)^2 + \dfrac{1}{4}\pi(15)^2 + \dfrac{1}{4}\pi(5)^2 = \dfrac{2125\pi}{4} \approx 1668,971$.

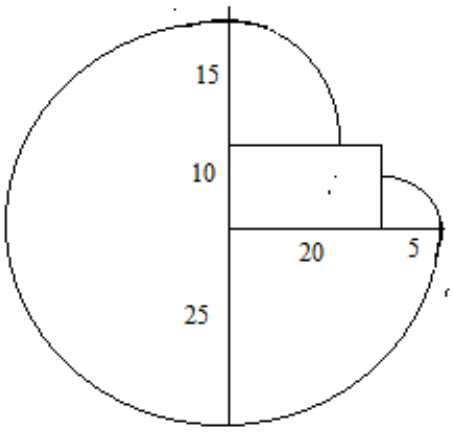

Рис. VI.1.6. Чертеж к задаче VI.1.2.4б)

в) Площадь выпаса равна 3/4 круга радиуса 50, плюс 1/4 круга радиуса 40, плюс 1/4 круга радиуса 30, минус общая часть двух последних кругов (см. рис. VI.1.7).

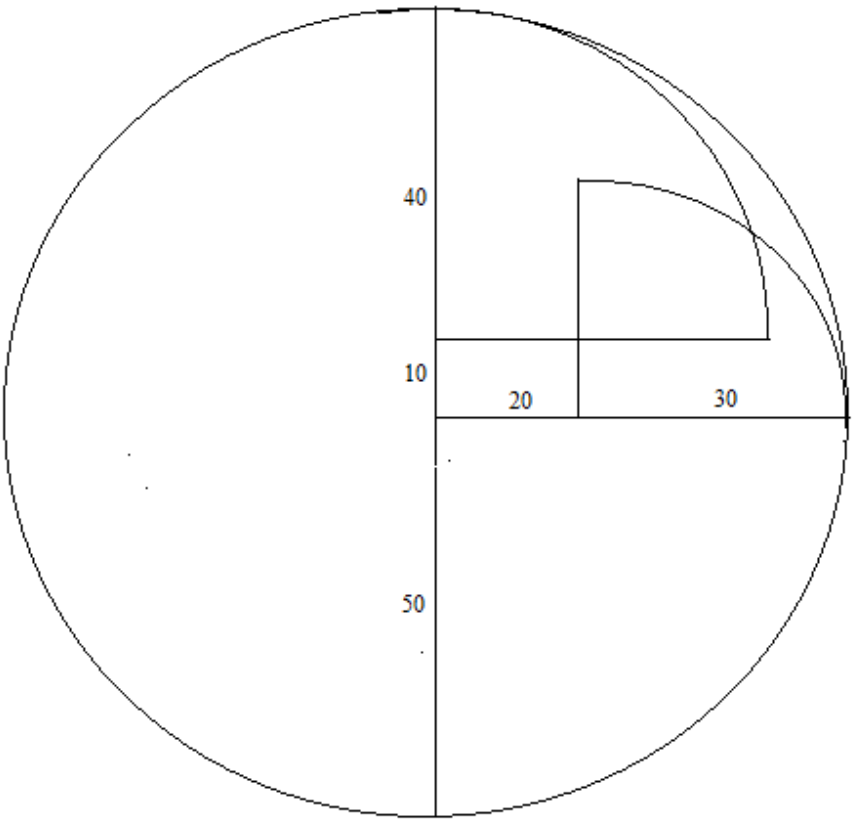

Рис. VI.1.7. Чертеж к задаче VI.1.2.4в)

Чтобы не искать общую часть, можно разбить искомую область на непересекающиеся части. Для этого соединим точку пересечения окружностей радиусов 40 и 30 с углами заграждения так, как показано на рисунке (см. рис. VI.1.8).

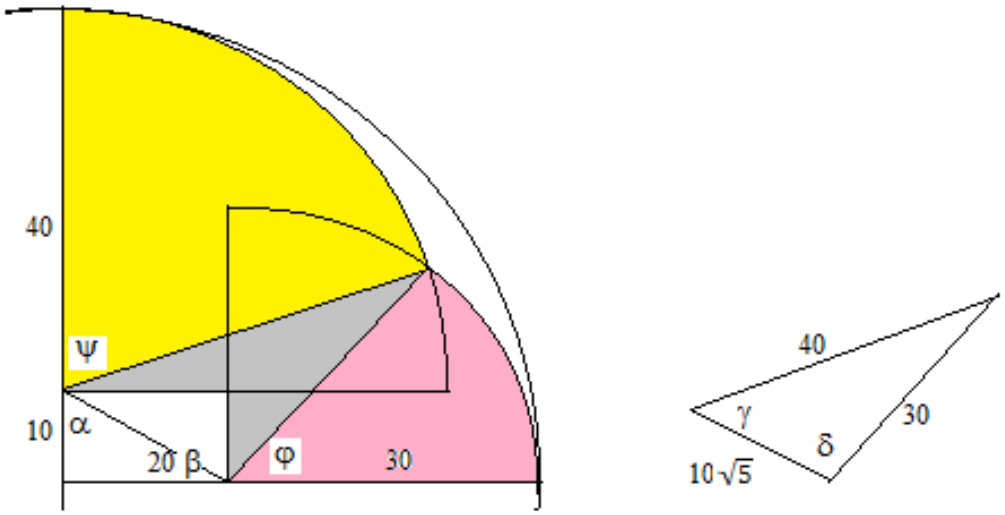

Рис. VI.1.8. Обозначения и дополнительные построения к задаче VI.1.2.4в)

Тогда искомая площадь, помимо 3/4 круга радиуса 50, содержит три непересекающиеся области: 1) четырехугольную площадь в виде «стрелочки», 2) сектор круга радиуса 40 с углом ψ, 3) сектор круга радиуса 30 с углом φ. Найдем вначале площадь «стрелочки». Проведем в прямоугольной загородке диагональ. Ее длина равна $\sqrt{10^2+20^2}=10\sqrt{5}$. Следовательно, в большом треугольнике, вынесенном в отдельный рисунок на рис. VI.1.8, стороны равны 40, 30 и $10\sqrt{5}$, и его площадь по формуле Герона равна

$$\sqrt{p(p-a)(p-b)(p-c)},$$

где p — полупериметр, а a,b,c — стороны. То есть площадь большого треугольника равна

$$\sqrt{(5\sqrt{5}+35)(35-5\sqrt{5})(5\sqrt{5}+5)(5\sqrt{5}-5)}=$$
$$=\sqrt{(35^2-25\cdot 5)(25\cdot 5-25)}=100\sqrt{11}.$$

Следовательно, площадь «стрелочки» равна

$$S_1=100\sqrt{11}-100\approx 231,6625.$$

Для нахождения площадей круговых секторов нужно знать их углы ψ и φ. Вначале вычислим косинусы углов α и β между диагональю прямоугольника загородки и его сторонами:

$$\cos\alpha=\frac{10}{10\sqrt{5}}=\frac{1}{\sqrt{5}},\ \cos\beta=\frac{20}{10\sqrt{5}}=\frac{2}{\sqrt{5}}.$$

Теперь посчитаем косинусы углов δ и γ в большом треугольнике по теореме косинусов $a^2+b^2-2ab\cos C=c^2$:

$$\cos\gamma=\frac{-30^2+40^2+(10\sqrt{5})^2}{2dd\cdot 40dd\cdot 10\sqrt{5}}=\frac{3}{2\sqrt{5}},$$

$$\cos\delta=\frac{-40^2+30^2+(10\sqrt{5})^2}{2dd\cdot 30dd\cdot 10\sqrt{5}}=\frac{-1}{3\sqrt{5}}.$$

Через углы δ и γ легко определить радианную меру углов ψ и φ круговых секторов:

$$\psi=\pi-\arccos\alpha-\arccos\gamma\approx 1,198962,$$

$$\varphi = \pi - \arccos\beta - \arccos\delta \approx 0,95752.$$

Тогда площади секторов находятся по формуле

$$S_2 = \frac{\psi}{2}r^2 = 959,1696,\ S_3 = \frac{\varphi}{2}r^2 \approx 430,8839.$$

А 3/4 площади круга радиуса 50 равны $S = \frac{3}{4}\pi(50)^2 \approx 5890,486$.

Итоговая площадь выпаса равна $S + S_1 + S_2 + S_3 \approx 7512,716$.

Ответ: а) $\frac{75\pi}{4} \approx 58,90486$ кв. ед., б) $\frac{2125\pi}{4} \approx 1668,971$ кв. ед., в) 7512,716 кв. ед.

Критерии оценивания

- Верно решены пункты а) и б) — 10 баллов.
- Верно решены пункты а) и б). Приведен правильный чертеж пункта в), который полностью не доведен до решения, но содержит некоторые верные шаги на пути к нахождению пересечения двух окружностей радиусов 40 и 30 — 20 баллов.
- Верно решены пункты а), б) и в) — 40 баллов.

Математика. 10–11 классы

Задача VI.1.3.1. (20 баллов)

Архитектор виртуальных миров Владимир тестирует разработанную компанией «Альтернативное будущее» цифровую землю — «Тридевятое царство». Ему известно, что среди прочих персонажей, населяющих виртуальную реальность, он может встретиться с двумя цифровыми аватарами самого себя. За полностью идентичным внешним видом скрываются некоторые различия двойников: один из них глаголет истину только в воскресенье, понедельник, вторник и среду и лжет с четверга по субботу, а второй — истину вещает с четверга по воскресенье и лжет с понедельника по среду. Согласно заданию квеста в рамках тестового прогона, Владимир находит ключ шифрования одного из цифровых аватаров и должен передать его владельцу. Если тестировщик не ошибется, то перейдет на следующий уровень игры, допустит ошибку — придется проходить дополнительные испытания. Наконец, в трехмерном пространстве Владимир находит одну из искомых моделей и спрашивает напрямик: «Чей это ключ?» Встреченный аватар ответил загадочно: «Истинный владелец ключа сегодня говорит правду,» — и Владимир вручил ему ключ шифрования. Каковы шансы на то, что Владимир ошибся, и ключом завладела не та компьютерная модель, которой он принадлежал, и теперь Владимиру придется тратить время на решение дополнительных задач?

Решение

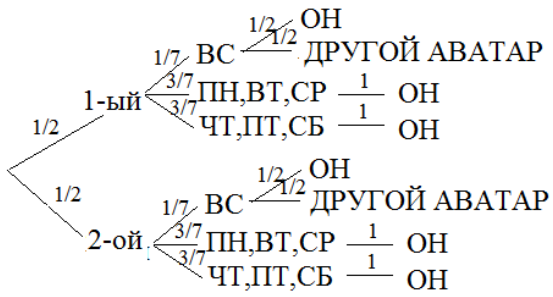

Рис. VI.1.9. Обозначения и дополнительные построения к задаче VI.1.3.1

Нарисуем дерево возможностей для этого случая. С вероятностью 1/2 Владимир встретил первого аватара и с вероятностью 1/2 — второго. Рассмотрим случай, когда Владимир встретил первого двойника. Если в день встречи было воскресенье (с вероятностью 1/7), то первый аватар говорит правду, и «истинный владелец ключа сегодня говорит правду», а в воскресенье правду говорят обе модели, следовательно, владельцем ключа может быть как первый аватар (с вероятностью 1/2), так и его двойник. Если в день встречи был понедельник, вторник или среда (с вероятностью 3/7), то первый аватар говорит правду, и «истинный владелец ключа сегодня говорит правду», а в эти дни правду говорит он сам, следовательно, владелец ключа он сам (с вероятностью 1). Если в день встречи был четверг, пятница или суббота, то первый аватар говорит ложь, и «истинный владелец ключа сегодня НЕ говорит правду», а в эти дни лжет он сам, следовательно, владелец ключа он сам (с вероятностью 1). Аналогично рассматривается случай встречи со вторым аватаром. Таким образом, вероятность что владелец ключа не говорящий, т. е. не «он сам» будет

$$\frac{1}{2}\cdot\frac{1}{7}\cdot\frac{1}{2}+\frac{1}{2}\cdot\frac{1}{7}\cdot\frac{1}{2}=\frac{1}{14}.$$

Ответ: шансы равны одному из четырнадцати, т. е. 7,14 процентов.

Критерии оценивания

- Правильно исследована логическая ситуация, вероятность найдена не верно — 10 балла.
- Правильно исследована логическая ситуация, верно найдена вероятность — 20 баллов.

Задача VI.1.3.2. (20 баллов)

Несколько монет, из которых мы знаем, что N орлов, разложены на столе. Игроку одевают виртуальные очки, в которые он видит стол, но не видит монет. Каким образом можно не видя разложить монеты на две группы так, чтобы число орлов в каждой было одинаково? Если игрок верно разложит монеты, программа проанализирует ситуацию на столе, и покажет монеты игроку!

Примечание: предположите, что на ощупь мы не можем отличить орла от решки.

Решение

Не видя мы можем: а) пересчитывать монеты, б) переворачивать монеты. Пусть окажется, что всего $N + M$ монет, т. е. решек M. Перевернем t монет. Пусть до переворота в группе из t монет было h орлов и $t - h$ решек. Тогда после переворота в этой группе будет $t - h$ орлов и h решек. В оставшейся группе будут $N + M - t$ монет, из которых $N - h$ орлов и $N + M - t - (N - h) = M - t + h$ решек. Пусть после переворота число орлов совпало в обеих группах: $t - h = N - h$. Тогда $t = N$. Следовательно, чтобы получить две группы монет с одинаковым количеством орлов, нужно перевернуть ровно N (столько, сколько было орлов) монет, и отложить их в одну группу. Оставшиеся монеты образуют вторую группу.

Ответ: нужно перевернуть ровно N (столько, сколько было орлов) монет, и отложить их в одну группу. Оставшиеся монеты образуют вторую группу.

Критерии оценивания

- Догадались, что в двух группах одинаковое количество орлов не обязано совпадать с половиной первоначального количества, или что монеты можно переворачивать, или приведена еще какая-нибудь полезная догадка, которая может привести к решению — 10 балла.
- Найдено решение, которое полностью соответствует условию задачи и математически обосновано — 20 баллов.

Задача VI.1.3.3. (20 баллов)

В дополненной реальности плотная цилиндровая подушка лежит у стены дома, и точка Q на окружности цилиндра отстает от стены на 2 см, а от пола — на 1 см. Длина подушки 20 см. Установить, поместится ли она в виртуальную коробку $21 \times 7 \times 7$ см.

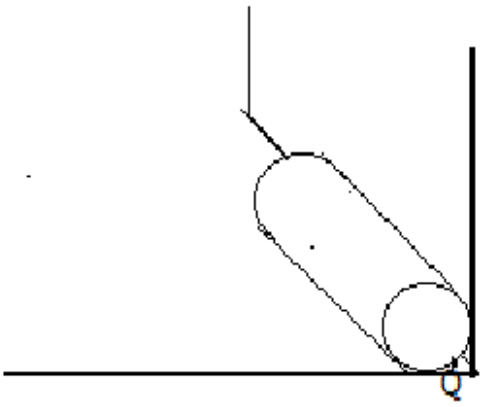

Рис. VI.1.10. Чертеж к задаче VI.1.3.3

Решение

1) 20 см $<$ 21 см, следовательно, по длине поместится.

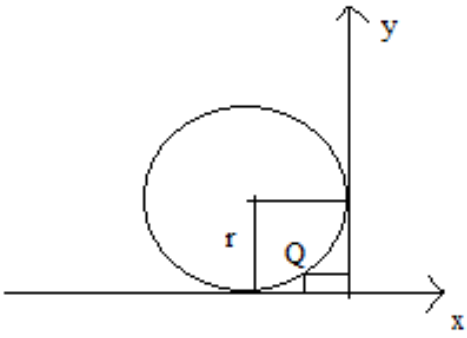

Рис. VI.1.11. Обозначения и дополнительные построения к задаче VI.1.3.3

2) Надо, чтобы $2r < 7$. Впишем окружность цилиндра в Декартову систему координат на плоскости так, что ось y — ось стены, а ось x — ось пола (см. рис. VI.1.11). Тогда центр окружности — точка $(-r, r)$, уравнение окружности $(x+r)^2+(y-r)^2 = r^2$. Так как точка $Q(-2, 1)$ принадлежит окружности,

$$(-2 + r)^2 + (1 - r)^2 = r^2.$$

Отсюда получаем квадратное уравнение: $r^2 - 6r + 5 = 0$, имеющее два корня: $r = 5$ и $r = 1$. Второй корень не подходит по построению, следовательно, радиус окружности цилиндра $r = 5$, и т. к. $2r > 7$, подушка не поместится в коробку.

Ответ: подушка не поместится в коробку, т. к. диаметр окружности цилиндра 10 см меньше 7 см.

Критерии оценивания

- Найдено верное уравнение, связывающее точку Q с радиусом окружности цилиндра, но решено не верно, или посторонний корень не отброшен — 10 баллов.
- Задача решена верно и полностью математически обоснована — 20 баллов.

Задача VI.1.3.4. (40 баллов)

В игре Minecraft построен конусовидный холм для обзора окрестностей. Самый короткий путь для полного обзора окрестности вокруг холма пролегает от точки A у подножья до точки B, расположенной на той же образующей конуса, что и точка A, но на 1 см выше точки A. Высота подъема на холм по образующей 6 см, радиус окружности основания холма — 2 см. По этому обзорному пути нужно вначале идти в гору, а потом спускаться к точке B. Найти длину спуска до B по этому пути. Примечание: постройте развертку конуса.

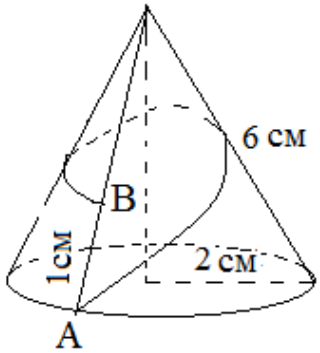

Рис. VI.1.12. Чертеж к задаче VI.1.3.4

Решение

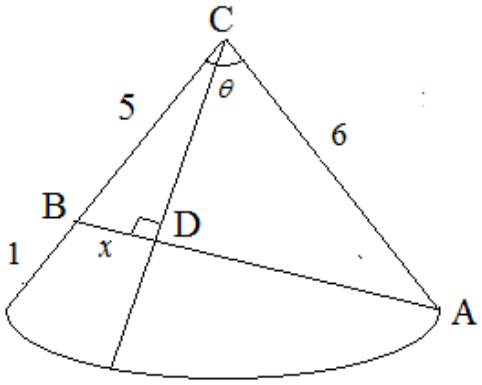

Рис. VI.1.13. Обозначения и дополнительные построения к задаче VI.1.3.4

Сделаем развертку конуса (рис. VI.1.13), разрезав его по одной из образующих. Развертка конуса представляет собой сектор круга, т. к. все образующие одной длины и могут быть радиусами круга. Длина дуги сектора равна длине окружности конуса, и следовательно равна $2\pi r = 4\pi$. Найдем центральный угол. Длина (L) дуги сектора равна произведению радиуса круга (R) и центрального угла, выраженного в радианах (θ): $L = R\theta$. Так как радиус круга равен образующей конуса, $4\pi = 6\theta$, $\theta = \dfrac{2\pi}{3}$.

Пусть точка A лежит в правом нижнем углу развертки, тогда точка B лежит на левой стороне развертки в 1 см от основания. Так как по условию путь между A и B кратчайший, на развертке он идет по прямой. Найдем длину отрезка AB. В $\triangle ABC$ по теореме косинусов

$$AB^2 = BC^2 + CA^2 - 2AB \cdot BC \cos\theta.$$

Найдем $\cos\theta = \cos\dfrac{2\pi}{3} = \cos(\pi - \dfrac{\pi}{3}) = -\cos\dfrac{\pi}{3} = -\dfrac{1}{2}$. Тогда $AB^2 = 36 + 25 + 30 = 91$, а $AB = \sqrt{91}$ — это длина всего пути.

Найдем длину спуска. Как найти, когда заканчивается подъем и начинается спуск? Если длина по образующей до вершины уменьшается, следовательно, мы поднимаемся, если увеличивается, мы спускаемся. Наименьшая длина до вершины будет, когда

образующая перпендикулярна AB. Рассмотрим $\triangle BCD$ и $\triangle ACD$. Пусть $BD = x$ — длина спуска. Тогда $AD = \sqrt{91} - x$ — длина подъема. Пусть $CD = h$. Тогда из двух прямоугольных треугольников имеем систему:

$$\begin{cases} (\sqrt{91} - x)^2 + h^2 = 36, \\ x^2 + h^2 = 25. \end{cases}$$

Вычтем из первого уравнения второе: $91 - 2\sqrt{91}x = 11$. Отсюда $x = \dfrac{40}{\sqrt{91}} \approx 4,193$ см.

Ответ: $\dfrac{40}{\sqrt{91}} \approx 4,193$ см.

Критерии оценивания

- Правильно построена развертка, верно обозначен путь AB — 5 баллов.
- Правильно построена развертка, верно обозначен путь AB, найдены длина дуги и центральный угол сектора — 10 баллов.
- Правильно найдена длина всего пути AB — 20 балла.
- Правильно рассчитана (возможно с округлением) длина спуска — 40 баллов.